INLG 2021

The 14th International Conference on Natural Language Generation

Proceedings of the Conference

20-24 September 2021
Aberdeen, Scotland, UK

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-954085-51-0

Preface

We are pleased to present the Proceedings of the 14th International Natural Language Generation Conference (INLG 2021). Due to the COVID-19 pandemic, this year the conference was primarily held online on 20-23 September 2021.

INLG 2021 was organised by the University of Aberdeen. The event took place under the auspices of the Special Interest Group on Natural Language Generation (SIGGEN) of the Association for Computational Linguistics (ACL).

The INLG conference is the main international forum for the discussion of all aspects of Natural Language Generation (NLG), including applications, evaluation, models, and resources.

The conference started on 20 Sept with a tutorial by David M. Howcroft on Crowdsourcing Experiments and Platforms.

The main conference took place over three days (21-23 Sep). We received 76 submissions (excluding Generation Challenges), of which 23 were accepted as long papers, 7 as short papers, and 1 as a demo paper.

Generation challenges is a set of shared tasks which are presented at INLG. This year results of 2 challenges were presented:

- Shared Task on Evaluating Accuracy in Generated Texts
- The ReproGen Shared Task on Reproducibility of Human Evaluations in NLG

The proceedings include a summary paper for each challenge, and 8 short papers describing submissions to the challenges. We also received 5 proposals for 5 new challenges for 2022, 4 of which were accepted and are included as short papers in the proceedings.

We were very grateful to our two keynote speakers:

- Tim Bickmore (Northeastern University, USA), who spoke on Health Counseling Dialogue Systems: Promise and Peril
- Natalie Schluter (Google Brain and IT University Copenhagen, Denmark), who spoke on Fresh Eyes on the Tough Problem of Automatic Summarisation

We are also grateful to the four members of our panel on What Users Want from Real-World NLG:

- Adam Sam (Monok)
- Ross Turner (Arria)
- Robert Weißgraeber (Ax Semantics)
- Michelle Zhou (Juji)

INLG would not have been possible without the generous financial support we received from our sponsors:

- ADAPT Centre
- Arria NLG
- Ax Semantics
- Google
- Hugging Face

We would also like to thank Abbey Conference Management for their hard work on delivering INLG 2021 online. Finally would also like to extend our gratitude to all speakers, (area) chairs and reviewers for their excellent work.

Anya Belz
Angela Fan
Ehud Reiter
Yaji Sripada

INLG 2021 Programme Chairs

Programme Chairs:

Anya Belz (ADAPT, Dublin City University, Ireland)
Angela Fan (Facebook, France)
Ehud Reiter (University of Aberdeen, Scotland, UK)
Yaji Sripada (University of Aberdeen, Scotland, UK)

Workshop Chair:

Emiel van Miltenburg (Tilburg University, Netherlands)

Publication Chair:

David M. Howcroft (Edinburgh Napier University, Scotland, UK)

Generation Challenge Chair:

Samira Shaikh (University of North Carolina Charlotte, USA)

Website:

Miruna Clinciu (Heriot-Watt University, Scotland, UK)

Local Organization Committee:

Ehud Reiter (University of Aberdeen, Scotland, UK)
Yaji Sripada (University of Aberdeen, Scotland, UK)

Invited Speakers:

Tim Bickmore (Northeastern University, USA)
Natalie Schluter (Google Brain and IT University Copenhagen, Denmark)

Panel Members:

Adam Sam (Monok)
Ross Turner (Arria)
Robert Weißgraeber (Ax Semantics)
Michelle Zhou (Juji)

Area Chairs:

Joan Byamugisha (IBM South Africa)
Ondřej Dušek (Charles University)
Thiago Castro Ferreira (Universidade Federal de Minas Gerais)

Albert Gatt (University of Malta)
Dimitra Gkatzia (Edinburgh Napier University)
Saad Mahamood (Trivago)
Lara Martin (University of Pennsylvania)
Shrimai Prabhumoye (Carnegie Mellon)
Natalie Schluter (University of Copenhagen)
Samira Shaikh (University of North Carolina Charlotte)
Sina Zarrieß (Friedrich Schiller Universität Jena)
Xingxing Zhang (Microsoft China)

Program Committee:

Malihe Alikhani (University of Pittsburgh)
Jose Alonso (University of Santiago de Compostela)
Jun Araki (Bosch Research)
Vidhisha Balachandran (Carnegie Mellon University)
David Bamutura (Chalmers University of Technology)
Jennifer Biggs (Defence Science and Technology Group)
Nadjet Bouayad-Agha (NLP Consultant)
Daniel Braun (TU Munich)
Gordon Briggs (U.S. Naval Research Laboratory)
Alberto Bugarín-Diz (University of Santiago de Compostela)
Jan Buys (University of Cape Town)
Michele Cafagna (University of Malta)
Guanyi Chen (Utrecht University)
Yagmur Gizem Cinar (Naver Labs Europe)
Elizabeth Clark (University of Washington)
Brian Davis (Dublin City University)
Rodrigo de Oliveira (Arria NLG)
Nina Dethlefs (University of Hull)
Martin Dominguez (Universidad Nacional de Cordoba)
Yuheng Du (Amazon)
Pablo Duboue (NLP Consultant)
Macarena Espinilla Estévez (University of Jaén)
Farhood Farahnak (Concordia University)
Cristina Garbacea (University of Michigan)
Lorenzo Gatti (Human Media Interaction, University of Twente)
Pablo Gervás (Universidad Complutense de Madrid)
Dimitra Gkatzia (Edinburgh Napier University)
Martijn Goudbeek (Tilburg University)
Ting Han (National Institute of Advanced Industrial Science and Technology)
Aki Härmä (Philips Research)
Sadid A. Hasan (CVS Health)
Raquel Hervas (University Complutense of Madrid)
Daphne Ippolito (University of Pennsylvana)
Amy Isard (University of Hamburg)
Takumi Ito (Tohoku University)
Harsh Jhamtani (Carnegie Mellon University)
Aditya Joshi (CSIRO)
Da Ju (Facebook)

Mihir Kale (Google)
Emiel Krahmer (Tilburg University)
Tatsuki Kuribayashi (Tohoku University)
Cyril Labbe (Université Grenoble Alpes)
Gerasimos Lampouras (Huawei Noah's Ark Lab)
Maurice Langner (Ruhr-Universität Bochum)
Lin Li (Qinghai Normal University)
Tianyu Liu (Peking University)
Elena Lloret (University of Alicante)
Saad Mahamood (Trivago)
Zola Mahlaza (University of Cape Town)
Aleksandre Maskharashvili (Ohio State University)
David McDonald (SIFT)
Antonio Valerio Miceli Barone (The University of Edinburgh)
Simon Mille (Pompeu Fabra University)
Diego Moussallem (Paderborn Univerisity)
Ryo Nagata (Konan University)
Daniel Paiva (Arria NLG)
Pablo Pérez De Angelis (TuQuejaSuma)
Paul Piwek (The Open University)
François Portet (Université Grenoble Alpes)
Sashank Santhanam (University of North Carolina Charlotte)
Lei Shu (University of Illinois at Chicago)
Marco Antonio Sobrevilla Cabezudo (Universidade de São Paulo)
Balaji Vasan Srinivasan (Adobe Research)
Somayajulu Sripada (University of Aberdeen)
Kristina Striegnitz (Union College)
Shahbaz Syed (Leipzig University)
Hiroya Takamura (Tokyo Institute of Technology)
Marc Tanti (University of Malta)
Mariët Theune (University of Twente)
Craig Thomson (University of Aberdeen)
Ross Turner (Arria NLG)
Kees van Deemter (Utrecht University)
Keith VanderLinden (Calvin College)
Stephen Wan (CSIRO)
Di Wang (Carnegie Mellon University)
Qingyun Wang (University of Illinois at Urbana-Champaign)
Robert Weißgraeber (Ax Semantics)
Michael White (Ohio State University)
Qiongkai Xu (The Australian National University)
Xinnuo Xu (Heriot-Watt University)
Jin-ge Yao (Peking University)
Zhirui Zhang (University of Science and Technology of China)
Yinhe Zheng (Alibaba)
Qingyu Zhou (Tencent)
Yanyan Zou (JD)
Ingrid Zukerman (Monash University)

Table of Contents

Conference Papers

Generation Challenge Papers

Conference Papers (continued)

Conference Program

Monday, 20 September 2021

Big Science Workshop (co-located event)

14:00–16:00 *Tutorial: Crowdsourcing Experiments & Platforms*
David M. Howcroft (Edinburgh Napier University)

Tuesday, 21 September 2021

Opening Plenary

12:00–12:10 *Welcome to INLG 2021*

12:10–13:00 *Invited Talk: Fresh Eyes on the Tough Problem of Automatic Summarisation*
Natalie Schluter (Google Brain ; ITU Copenhagen)

13:20–14:40 Oral Session 1: Models & Questions

Generating Diverse Descriptions from Semantic Graphs
Jiuzhou Han, Daniel Beck and Trevor Cohn

Neural Methodius Revisited: Do Discourse Relations Help with Pre-Trained Models Too?
Aleksandre Maskharashvili, Symon Stevens-Guille, Xintong Li and Michael White

Exploring Input Representation Granularity for Generating Questions Satisfying Question-Answer Congruence
Madeeswaran Kannan, Haemanth Santhi Ponnusamy, Kordula De Kuthy, Lukas Stein and Detmar Meurers

Towards Zero-Shot Multilingual Synthetic Question and Answer Generation for Cross-Lingual Reading Comprehension
Siamak Shakeri, Noah Constant, Mihir Kale and Linting Xue

Tuesday, 21 September 2021 (continued)

14:40–15:40 Mid-afternoon Break

15:40–16:40 Poster Session 1

Chefbot: A Novel Framework for the Generation of Commonsense-enhanced Responses for Task-based Dialogue Systems
Carl Strathearn and Dimitra Gkatzia

Predicting Antonyms in Context using BERT
Ayana Niwa, Keisuke Nishiguchi and Naoaki Okazaki

Examining Covert Gender Bias: A Case Study in Turkish and English Machine Translation Models
Chloe Ciora, Nur Iren and Malihe Alikhani

WeaSuL: Weakly Supervised Dialogue Policy Learning: Reward Estimation for Multi-turn Dialogue
Anant Khandelwal

Multi-Sentence Knowledge Selection in Open-Domain Dialogue
Mihail Eric, Nicole Chartier, Behnam Hedayatnia, Karthik Gopalakrishnan, Pankaj Rajan, Yang Liu and Dilek Hakkani-Tur

Self-Training for Compositional Neural NLG in Task-Oriented Dialogue
Xintong Li, Symon Stevens-Guille, Aleksandre Maskharashvili and Michael White

Generating Racing Game Commentary from Vision, Language, and Structured Data
Tatsuya Ishigaki, Goran Topic, Yumi Hamazono, Hiroshi Noji, Ichiro Kobayashi, Yusuke Miyao and Hiroya Takamura

Tuesday, 21 September 2021 (continued)

17:00–18:00 Birds of a Feather Sessions

18:00–19:00 Social Hour

Wednesday, 22 September 2021

12:00–13:20 Oral Session 2: NLG Tasks, Evaluation, & Explanation

Explaining Decision-Tree Predictions by Addressing Potential Conflicts between Predictions and Plausible Expectations
Sameen Maruf, Ingrid Zukerman, Ehud Reiter and Gholamreza Haffari

Formulating Neural Sentence Ordering as the Asymmetric Traveling Salesman Problem
Vishal Keswani and Harsh Jhamtani

Underreporting of errors in NLG output, and what to do about it
Emiel van Miltenburg, Miruna Clinciu, Ondřej Dušek, Dimitra Gkatzia, Stephanie Inglis, Leo Leppänen, Saad Mahamood, Emma Manning, Stephanie Schoch, Craig Thomson and Luou Wen

What can Neural Referential Form Selectors Learn?
Guanyi Chen, Fahime Same and Kees van Deemter

13:40–14:40 Poster Session 2

HI-CMLM: Improve CMLM with Hybrid Decoder Input
Minghan Wang, GUO Jiaxin, Yuxia Wang, Yimeng Chen, Su Chang, Daimeng Wei, Min Zhang, Shimin Tao and Hao Yang

Using BERT for choosing classifiers in Mandarin
Jani Järnfors, Guanyi Chen, Kees van Deemter and Rint Sybesma

Enriching the E2E dataset
Thiago Castro Ferreira, Helena Vaz, Brian Davis and Adriana Pagano

Goal-Oriented Script Construction
Qing Lyu, Li Zhang and Chris Callison-Burch

Single Example Can Improve Zero-Shot Data Generation
Pavel Burnyshev, Valentin Malykh, Andrey Bout, Ekaterina Artemova and Irina Piontkovskaya

SAPPHIRE: Approaches for Enhanced Concept-to-Text Generation
Steven Feng, Jessica Huynh, Chaitanya Prasad Narisetty, Eduard Hovy and Varun Gangal

Contextualizing Variation in Text Style Transfer Datasets
Stephanie Schoch, Wanyu Du and Yangfeng Ji

14:40–15:40 Mid-afternoon Break

15:40–16:20 Generation Challenges: Results

Generation Challenges: Results of the Accuracy Evaluation Shared Task
Craig Thomson and Ehud Reiter

The ReproGen Shared Task on Reproducibility of Human Evaluations in NLG: Overview and Results
Anya Belz, Anastasia Shimorina, Shubham Agarwal and Ehud Reiter

16:30–17:10 Generation Challenges: Posters from the Accuracy Shared Task and ReproGen

Text-in-Context: Token-Level Error Detection for Table-to-Text Generation
Zdeněk Kasner, Simon Mille and Ondřej Dušek

Shared Task in Evaluating Accuracy: Leveraging Pre-Annotations in the Validation Process
Nicolas Garneau and Luc Lamontagne

Automatic Verification of Data Summaries
Rayhane Rezgui, Mohammed Saeed and Paolo Papotti

Grounding NBA Matchup Summaries
Tadashi Nomoto

Reproducing a Comparison of Hedged and Non-hedged NLG Texts
Saad Mahamood

Another PASS: A Reproduction Study of the Human Evaluation of a Football Report Generation System
Simon Mille, Thiago Castro Ferreira, Anya Belz and Brian Davis

A Reproduction Study of an Annotation-based Human Evaluation of MT Outputs
Maja Popović and Anya Belz

TUDA-Reproducibility @ ReproGen: Replicability of Human Evaluation of Text-to-Text and Concept-to-Text Generation
Christian Richter, Yanran Chen and Steffen Eger

17:20–18:00 Generation Challenges: New Challenges

DialogSum Challenge: Summarizing Real-Life Scenario Dialogues
Yulong Chen, Yang Liu and Yue Zhang

Quality Evaluation of the Low-Resource Synthetically Generated Code-Mixed Hinglish Text
Vivek Srivastava and Mayank Singh

Shared Task on Feedback Comment Generation for Language Learners
Ryo Nagata, Masato Hagiwara, Kazuaki Hanawa, Masato Mita, Artem Chernodub and Olena Nahorna

The SelectGen Challenge: Finding the Best Training Samples for Few-Shot Neural Text Generation
Ernie Chang, Xiaoyu Shen, Alex Marin and Vera Demberg

Wednesday, 22 September 2021 (continued)

18:00–19:00 SIGGEN Business Meeting

Thursday, 23 September 2021

12:00–13:00 Oral Session 3: Emotions & User Adaptation

Affective Decoding for Empathetic Response Generation
Chengkun Zeng, Guanyi Chen, Chenghua Lin, Ruizhe Li and Zhi Chen

Controllable Sentence Simplification with a Unified Text-to-Text Transfer Transformer
Kim Cheng SHEANG and Horacio Saggion

SEPRG: Sentiment aware Emotion controlled Personalized Response Generation
Mauajama Firdaus, Umang Jain, Asif Ekbal and Pushpak Bhattacharyya

13:20–14:40 Panel Discussion

Adam Sam (*Monok*), Ross Turner (*Arria*), Robert Weißgraber (*Ax Semantics*), & Michelle Zhou (*Juji*)

14:40–15:40 Mid-afternoon Break

Thursday, 23 September 2021 (continued)

15:40–16:40 Poster Session 3

Biomedical Data-to-Text Generation via Fine-Tuning Transformers
Ruslan Yermakov, Nicholas Drago and Angelo Ziletti

Decoding, Fast and Slow: A Case Study on Balancing Trade-Offs in Incremental, Character-level Pragmatic Reasoning
Sina Zarrieß, Hendrik Buschmeier, Ting Han and Simeon Schüz

GraphPlan: Story Generation by Planning with Event Graph
Hong Chen, Raphael Shu, Hiroya Takamura and Hideki Nakayama

BERT-based distractor generation for Swedish reading comprehension questions using a small-scale dataset
Dmytro Kalpakchi and Johan Boye

Exploring Structural Encoding for Data-to-Text Generation
Joy Mahapatra and Utpal Garain

Attention Is Indeed All You Need: Semantically Attention-Guided Decoding for Data-to-Text NLG
Juraj Juraska and Marilyn Walker

Closing Plenary

17:00–17:50 *Invited Talk: Health Counseling Dialogue Systems: Promise and Peril*
Tim Bickmore (Northeastern University)

17:50–18:00 *Closing Remarks*

Friday, 24 September 2021

Excursion to Dunottar Castle

Generating Diverse Descriptions from Semantic Graphs

Jiuzhou Han **Daniel Beck** **Trevor Cohn**
School of Computing and Information Systems
The University of Melbourne, Australia
jiuzhouh@foxmail.com, {d.beck,trevor.cohn}@unimelb.edu.au

Abstract

Text generation from semantic graphs is traditionally performed with deterministic methods, which generate a unique description given an input graph. However, the generation problem admits a range of acceptable textual outputs, exhibiting lexical, syntactic and semantic variation. To address this disconnect, we present two main contributions. First, we propose a stochastic graph-to-text model, incorporating a latent variable in an encoder-decoder model, and its use in an ensemble. Second, to assess the diversity of the generated sentences, we propose a new automatic evaluation metric which jointly evaluates output diversity and quality in a multi-reference setting. We evaluate the models on WebNLG datasets in English and Russian, and show an ensemble of stochastic models produces diverse sets of generated sentences, while retaining similar quality to state-of-the-art models.

1 Introduction

Semantic graphs are an integral part of knowledge bases that integrate and store information in a structured and machine-accessible way (van Harmelen et al., 2008). They are usually limited to specific domains, describing concepts, entities and their relationships in the real world. Generating descriptions from semantic graphs is an important application of Natural Language Generation (NLG) and can be framed in a *graph-to-text* transduction approach.

In recent years, approaches to graph-to-text generation can be broadly categorised into two groups. The first uses a sequence-to-sequence model (Trisedya et al., 2018; Konstas et al., 2017; Ferreira et al., 2019): the key step in this approach is to linearise the input graph to a sequence. Sequence-to-sequence models have been proved to be effective for tasks like question answering (Yin et al., 2016), text summarisation (Nallapati et al., 2016), and constituency parsing (Vinyals et al., 2015). However, when dealing with graph inputs, this method does not take full advantage of the graph structure. Another approach is to handle the graph directly, using a graph-to-sequence model (Ribeiro et al., 2020; Beck et al., 2018; Zhao et al., 2020). This approach has been recently widely adopted as it shows better performance for generating text from graphs (Xu et al., 2018).

The models used in previous work are *deterministic*: given the same input graph, they will always generate the same text (assuming a deterministic decoding algorithm is used). However, it is widely known that many graphs admit multiple valid descriptions. This is evidenced by the presence of *multiple references* in datasets such as WebNLG (Gardent et al., 2017a,b) and it is a common phenomenon in other generation tasks such as machine translation and image captioning. In this work, we propose to use models that generate *sets of descriptions* instead of a single one. In particular, we develop *stochastic* models with latent variables that capture *diversity* aspects of semantic graph descriptions, such as lexical and syntactic variability. We also propose a novel evaluation methodology that combines quality and diversity into a single score, in order to address caveats of previously proposed diversity metrics. Our findings show that stochastic models perform favourably when generating sets of descriptions, without sacrificing the quality of state-of-the-art architectures.

2 Related Work

Graph-to-sequence Models Standard graph-to-sequence models have two main components: a graph encoder and a sequence decoder. The encoder learns the hidden representation of the input graph and the decoder generates text based on this representation. Different graph-to-sequence mod-

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 1–11,
Aberdeen, Scotland, UK, 20-24 September 2021.

els vary mainly in the graph encoders.

Marcheggiani and Perez-Beltrachini (2018) proposed an encoder based on Graph Convolutional Networks (Kipf and Welling, 2017, GCNs), which directly exploit the input structure. Similar to Convolutional Neural Networks (LeCun et al., 1998), GCN layers can be stacked, resulting in representations that take into account non-adjacent, long-distance neighbours. Beck et al. (2018) used Gated Graph Neural Networks (Li et al., 2016) by extending networks on graph architectures with gating mechanisms, similar to Gated Recurrent Units (Cho et al., 2014, GRUs). Koncel-Kedziorski et al. (2019) proposed Graph Transformer Encoder by extending Transformers (Vaswani et al., 2017) to graph-structured inputs, based on the Graph Attention Network (Velickovic et al., 2017, GAT) architecture. This graph encoder generates node embeddings by attending over its neighbours through a self-attention strategy. Ribeiro et al. (2020) propose new models to encode an input graph with both global and local node contexts. To combine these two node representations together, they make a comparison between a cascaded architecture and a parallel architecture.

Latent Variable Models Within neural networks, a standard approach for generative models with latent variables is the Variational Autoencoder (VAE) (Kingma and Welling, 2014). The generative process is represented as: $p_\theta(x, z) = p_\theta(x \mid z)p_\theta(z)$, where $p_\theta(z)$ is the prior from which the latent variable is drawn, $p_\theta(x \mid z)$ is the likelihood of data point x conditioned on the latent variable z, typically calculated using a deep non-linear neural network, and θ denotes the model parameters.

Bowman et al. (2016) proposed a pioneering variational autoencoder for text generation to explicitly learn the global features using a continuous latent variable. They adapt the VAE to text data using an LSTM (Hochreiter and Schmidhuber, 1997) for both the encoder and the decoder, using a Gaussian prior to build a sequence autoencoder. This architecture can be extended to *conditional* tasks (when there is an input guiding the generation). Zhang et al. (2016) proposed an end-to-end variational model for Neural Machine Translation (NMT), using a continuous latent variable to capture the semantics in source sentences and guide the translation process. Schulz et al. (2018) proposed a more expressive word-level machine translation model incorporating a chain of latent variables, modelling lexical and syntactic variation in parallel corpora.

Diversity in Neural Networks and Generation Variational latent variable models are commonly employed when there is a need for generating diverse outputs. This is achieved by sampling from the latent variable every time a new output is required. One can also use a standard deterministic model and sample from the decoder distributions instead but this tends to decrease the quality of the generated outputs. Here we review a few common techniques to address this issue.

Dropout (Srivastava et al., 2014) is a regularisation method used to prevent overfitting in neural networks. At training time, it masks random parameters in the network at every iteration. Dropout can also be employed in the testing phase, during generation. This idea was first proposed by Gal and Ghahramani (2016) and it is also called Monte Carlo (MC) dropout. Because MC dropout disables neurons randomly, the network will have different outputs every generation, which can make a deterministic model generate different outputs.

Another technique to generate diverse outputs is *ensemble* learning. Typically, they are employed to prevent overfitting but they can also be used to generate diverse outputs. The idea is for each individual model in the ensemble to generate its own output. This approach can be very useful as each model tends to provide different optimal solutions in the network parameter space. This property has shown to benefit uncertainty estimation in deep learning (Lakshminarayanan et al., 2017). It can also be used both with deterministic and stochastic models, a property we exploit in our experiments.

3 Stochastic Graph-to-Sequence Model

In this section we introduce the proposed approach to generate diverse descriptions from semantic graphs. We start from the state-of-the-art model of Ribeiro et al. (2020), which is a deterministic graph-to-sequence architecture. Then we incorporate a latent variable and a variational training procedure to this model, in order to turn the model stochastic. This latent variable aims at capturing linguistic variations in the descriptions and is responsible for increasing the diversity at generation time. The architecture is shown in Figure 1.

3.1 Graph Encoder and Text Decoder

The encoder is similar to Ribeiro et al. (2020), consisting of a *global* and a *local* subencoder. The

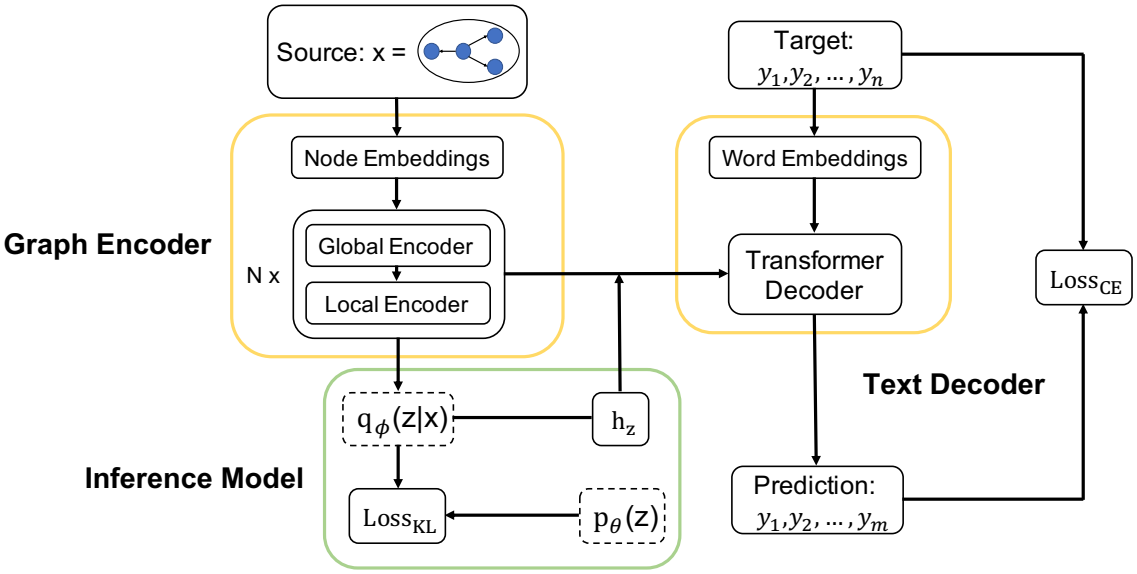


Figure 1: Proposed stochastic graph-to-sequence model architecture.

global encoder considers a wide range of contexts but it ignores the graph topology by considering each node as if it were connected to all the other nodes in the graph. The local encoder learns the hidden representation of each node on the basis of its neighbour nodes, which exploits the graph structure effectively. Combining both global and local node aggregations, this encoder can learn better contextualised node embeddings. The global encoding strategy is mainly based on the Transformer architecture (Vaswani et al., 2017), using a self-attention mechanism to calculate node representations of all nodes in the graph. The local encoding strategy adopts a modified version of Graph Attention Network (Velickovic et al., 2017) by adding relational weights to calculate the local node representations.

The decoder is also based on a *transformer* architecture. In our model, the input of the decoder is the contextualised node embeddings h_x concatenated with the hidden state of the latent variable h_z, which can be represented as $[h_x; h_z]$. Following Ribeiro et al. (2020), we also use beam search with length penalty (Wu et al., 2016) to encourage the model to generate longer sentences.

3.2 Inference Model

Here is where we introduce a latent Gaussian variable z, which together with the input graph x, guides the generation process. With this, the conditional probability of sentence y given x is

$$p(y|x) = \int_z p(y|z,x)p(z|x)dz.$$

The posterior inference in this model is intractable. Following previous work (Bowman et al., 2016; Kingma and Welling, 2014), we employ neural networks to fit the posterior distribution, to make the inference tractable. We regard the posterior as a diagonal Gaussian $\mathcal{N}\left(\mu, \text{diag}\left(\sigma^2\right)\right)$. The mean μ and variance σ^2 are parameterised with feed-forward neural networks (FFNNs), using the reparametrisation trick (Bowman et al., 2016; Kingma and Welling, 2014) of the Gaussian variables. It reparameterises the latent variable z as a function of mean μ and variance σ:

$$z = \mu + \sigma \odot \epsilon \quad \epsilon \sim \mathcal{N}(0, \text{I}),$$

where ϵ is a standard Gaussian variable which plays the role of introducing noises, and $\odot$ denotes element-wise multiplication. The reparametrisation trick enables back-propagation in optimisation process with Stochastic Gradient Descent (SGD). Then we transform the latent variable z into its hidden state h_z through another FFNN.

The training objective encourages the model to keep its posterior distributions $q(z \mid \mathbf{x})$ close to a prior $p(z)$ that is a standard Gaussian $\mathcal{N}(\mu = 0, \sigma = 1)$. The loss function of the stochastic conditional model can be defined as

$$\mathcal{L}(\phi, \theta; \mathbf{x}, \mathbf{y}) = -\mathbb{E}_{z \sim q_\phi(z|\mathbf{x})}\left[\log p_\theta\left(\mathbf{y} \mid z, \mathbf{x}\right)\right] + \text{KL}\left(q_\phi\left(z \mid \mathbf{x}\right) \| p(z)\right).$$

The first term is the expected negative log-likelihood of data which is called reconstruction loss or cross-entropy loss. It forces the model to learn to reconstruct the data. The second term is the KL divergence which acts as a regulariser. By minimising the KL term, we want to make the approximate posterior stay close to the prior. We use SGD to optimise the loss function.

3.3 Optimisation

As shown above, the stochastic model objective comprises two terms reconstruction and KL regularisation. The KL divergence term will be non-zero and the cross-entropy term will be relatively small if the model encodes task-relevant information in the latent variable z. A difficulty of training is that the KL term tends to zero, causing the model to ignore z. This makes the model deterministic. This phenomenon is also known as the *KL collapse* or *KL vanishing problem* (Lucas et al., 2019). We adopt the *KL Threshold* method (Pagnoni et al., 2018) to alleviate this issue. In this approach, we introduce a threshold ζ into the loss function to control the KL term. A large KL term means the latent variable learns much information. By setting a threshold, we can force the model to take at least a fixed KL regularisation cost. In our experiments, we set the threshold ζ as 10. The new loss function can be represented as

$$\mathcal{L}(\phi, \theta; \mathbf{x}, \mathbf{y}) = -\mathbb{E}_{z \sim q_\phi(z|\mathbf{x})} \left[\log p_\theta \left(\mathbf{y} \mid z, \mathbf{x}\right)\right] + \max\left(\mathrm{KL}\left(q_\phi\left(z \mid \mathbf{x}\right) \| p(z)\right), \zeta\right).$$

4 Joint Evaluation of Diversity and Quality

Addressing diversity in language generation is a recent topic that attracted attention in particular in image captioning. This led to the development of metrics that aim at measuring the diversity of a set of sentences, such as Self-BLEU (Zhu et al., 2018). However, these metrics are based only on the generated output space, ignoring the references in the gold standard. This lead to spurious measurements, such as unconditional language models having excellent performance according to these metrics, even though they have no practical use as they ignore the input.

To address these caveats, we propose a new evaluation procedure that assesses diversity and quality *jointly*. Our key insight (and assumption) is based on using the reference set as a gold standard for both aspects. Given a graph, the set of references acts as the "end goal", containing high-quality descriptions with sufficient levels of diversity.[1] We call this procedure **Multi-Score (MS)**.

The idea behind Multi-Score is shown pictorially in Figure 2. In this example, we have a single instance with three references and three predicted descriptions generated by a model. Given a sentence-level quality metric we can calculate it among *all possible pairs* between each prediction and reference, obtaining a weighted bipartite graph. We then solve the respective *maximum matching problem* for this bipartite graph and take the average weight of the edges corresponding to the optimal matching. We show the full procedure to calculate Multi-Score in Algorithm 1.

Algorithm 1 Multi-Score procedure

```
function MULTI-SCORE(o: outputs, r: references, M: sentence-level metric)
    G ← 0                                   ▷ initialise graph
    for i ← 0 to len(o) do                  ▷ fill graph
        for j ← 0 to len(r) do
            G(i, j) ← M(o[i], r[j])
    match ← MAXMATCH(G)                     ▷ stores edges
    score ← 0
    for edge ∈ match do
        score ← score + edge.weight
    return score / len(match)
                                            ▷ returns average weight
```

For the example in Figure 2, the optimal matching (shown in red) matches prediction 1 with output 2, prediction 2 with output 3 and prediction 3 with output 1. From this, the resulting Multi-Score is: $(56+50+58)/3 = \mathbf{54.67}$. The matching problem MAXMATCH can be solved using the Hungarian Algorithm (Kuhn, 2010) in $O(n^3)$ time, where n is the number of nodes in the bipartite graph. This makes the procedure efficient for reference set sizes found in standard datasets.

As a metric, Multi-Score has a number of desirable properties:

- As long as the sentence-level metric has an upper bound (which is the case of most standard automatic evaluation metrics), if the set of predictions is exactly equal to the references, then MS will give the maximum score.

[1] We discuss limitations of this assumption in Section 7.

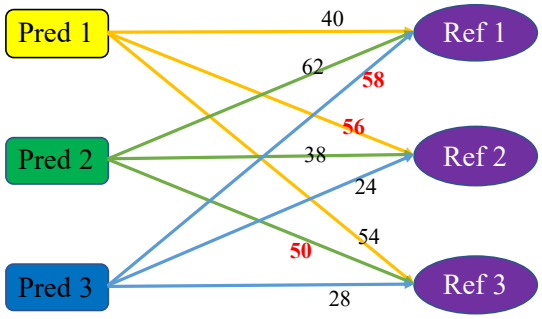


Figure 2: An example of calculating Multi-Score. The three "Pred" nodes on the left side represent three predicted descriptions while the three "Ref" nodes on the right side represent three references. The weight of each edge corresponds to the sentence-level quality score of this prediction-reference pair. The highlighted scores are the ones corresponding to the maximal matching, which are then used to calculate the MS metric. Other scores are ignored.

- If the outputs are diverse but unrelated to the references (as in an unconditional LM), MS will penalise the output because the underlying quality values will be low.
- If the outputs are high-quality but not diverse (typical of an n-best list in a deterministic model), MS will penalise the output due to the assignment constraint. One of the outputs will have a high-quality value but the others will have a low-quality value because they will be forced to match other references.
- Finally, MS can be used with any sentence-level quality metric, making it easily adaptable to any developments in better quality metrics, as well as other generation tasks.

5 Experimental Settings

5.1 Dataset

We evaluate the models using datasets from the WebNLG shared tasks (Gardent et al., 2017a,b). The data is composed of data-text pairs where the data is a set of RDF triples extracted from DBpedia and the text is the verbalisation of these triples. For each graph, there may be multiple descriptions. In our experiments, we assume a reference set of size 3 for each input, as most graphs in both datasets have three reference descriptions.

English WebNLG 2017 This dataset contains 18102 training, 872 development and 971 test data-text pairs. Entities are classified into 15 distinct categories (Astronaut, University, Monument, Building, ComicsCharacter, Food, Airport, SportsTeam, WrittenWork, Athlete, Artist, City, MeanOfTransportation, CelestialBody, Politician).

Russian WebNLG 2020 The Russian dataset comprises 16571 training, 790 development and 1102 test data-text pairs. This dataset has 9 distinct categories (Airport, Astronaut, Building, CelestialBody, ComicsCharacter, Food, Monument, SportsTeam, and University).

5.2 Preprocessing

Levi Graph Transformation To decrease the number of parameters and avoid parameter explosion, we follow previous work and use a Levi Graph Transformation (Ribeiro et al., 2020; Beck et al., 2018). This transformation creates new relation nodes from relational edges between entities, which explicitly represents the relations between an original node and its neighbour edges.

Byte Pair Encoding Following previous work (Ribeiro et al., 2020), we employ Byte Pair Encoding (BPE) to split entity words into frequent characters or character sequences which are subword units. After the BPE operations, some nodes in the graph are split to subwords. Likewise, we also split the target descriptions using BPE.

5.3 Models

All models are able to generate sets of descriptions: we generate three sentences per graph as this matches the number of available references. For the proposed stochastic models, we generate each sentence by sampling a new value for the latent variable. For the deterministic models, we use different decoding strategies to generate these sets.

Top-3 Beam Search Beam Search is the standard algorithm to obtain a sentence from deterministic models, by selecting the output with (approximate) highest probability. In Top-3 Beam Search, we choose the top-3 generated sentences from the final candidate list.

Total Random Sampling Random Sampling (Ippolito et al., 2019) generates a sentence from left to right sampling the next token from all possible candidates until the end-of-sequence symbol is generated. Because each token is *sampled* from

the distribution over next tokens given the previous ones, this method generates different outputs each time it generates a new description.

Top-3 Random Sampling In this approach, we still use Random Sampling but modify it slightly while generating the next token. Instead of sampling the next token from all possible candidates, the model samples the next token from the top-3 most likely candidates (Ippolito et al., 2019).

MC Dropout We employ MC dropout to the deterministic model and keep the dropout rate in the testing phase and training phase the same. It disables neurons randomly at decoding time, resulting in different outputs at each generation.

Ensemble Finally, we create an ensemble of three independently-trained deterministic models, whereby we select the most likely sentence from each model using Beam Search. These sentences then form the output set from the ensemble. Since this is a general strategy, *we also apply it to the stochastic model* as another point of comparison in our experiments.

6 Results

We assess each model on the test set of English and Russian datasets respectively and report the quality and diversity results. The quality evaluation scores (BLEU: Papineni et al. (2002), CHRF++: Popovic (2017)) are calculated based on the average score of the three outputs. We report the original BLEU and CHRF++ score to show the quality of the generated sentences from each model. The diversity evaluation scores (Self-BLEU, Multi-Score) are computed using the three outputs. As we describe in Section 4, our proposed diversity evaluation metrics require a sentence-level quality evaluation metric to compute the score of two sentences. We adopt sentence-level BLEU and CHRF++ and refer to their corresponding Multi-Score versions as MS-BLEU and MS-CHRF.

Table 1 shows the quality results on both English and Russian datasets. As expected, the two random sampling methods do not show good quality performance. For English data, our stochastic models perform on par with previous work and have comparable quality with deterministic models. The trends for English and Russian data are similar but Russian has lower scores in general.

The diversity scores of these two datasets are shown in Table 2. *Total random sampling* has the lowest Self-BLEU on two datasets, as expected, but it also has the worst quality. On the other hand, with our new metrics, the stochastic ensemble model gives the best results on both English and Russian datasets, showing high diversity without compromising quality.

6.1 Error Analysis

To further assess the quality of the generated sentences from each model, we perform a manual error analysis in a subset of the English test data. We randomly selected five input graphs, generating 15 sentences for each model (as we generate 3 sentences for each graph). Given we analysed five models, this gives a total of 75 sentences for our analysis. We observed three common mistakes from the outputs:

- **Syntax/Spelling Mistake:** There are grammar mistakes or spelling mistakes.
- **Lack of Information:** The information in the graph is not fully realised in the description.
- **Information Redundancy:** Some information in the sentence is repeated.

We calculate the rates of each model making different types of mistakes and report the results in Table 3. The results show that total random sampling makes the most mistakes among all models and most of them are syntax or spelling mistakes. *Top-3 random sampling* and *MC dropout* make the same percentage of total mistakes. The former makes almost half of the total information redundancy mistakes while the latter makes the most lack of information mistakes. Top-3 beam search makes fewer mistakes than the other three models and it does not make information redundancy mistakes in our evaluated test cases.

As for ensemble-based models, both deterministic and stochastic ensembles make the fewest total mistakes among all models. This is in line with the results obtained from automatic quality metrics. In particular, the deterministic ensemble does not make any syntax or spelling mistakes in the evaluated test cases. The stochastic ensemble also shows good performance with regard to the quality of the generated sentences, which has a low error rate for all types of mistakes.

In general, the diverse outputs generated by our proposed model tend to have comparable quality to the outputs from the best baseline model. However,

	English		Russian	
	BLEU↑	**CHRF++↑**	**BLEU↑**	**CHRF++↑**
Deterministic Models				
Top-3 beam search	62.69	74.48	52.50	64.76
Total random sampling	49.01	66.35	40.62	57.06
Top-3 random sampling	56.62	71.16	46.91	61.45
MC dropout	59.10	71.57	47.97	61.41
Ensemble	**63.31**	**74.52**	**53.60**	**65.30**
Stochastic Models				
Single model	62.81	74.12	52.45	64.43
Ensemble	62.88	74.25	52.60	64.38
Previous Work				
Melbourne (Gardent et al., 2017b)	54.52	70.72	-	-
Adapt (Gardent et al., 2017b)	60.59	76.01	-	-
CGE-LW (Ribeiro et al., 2020)	63.69	76.66	-	-

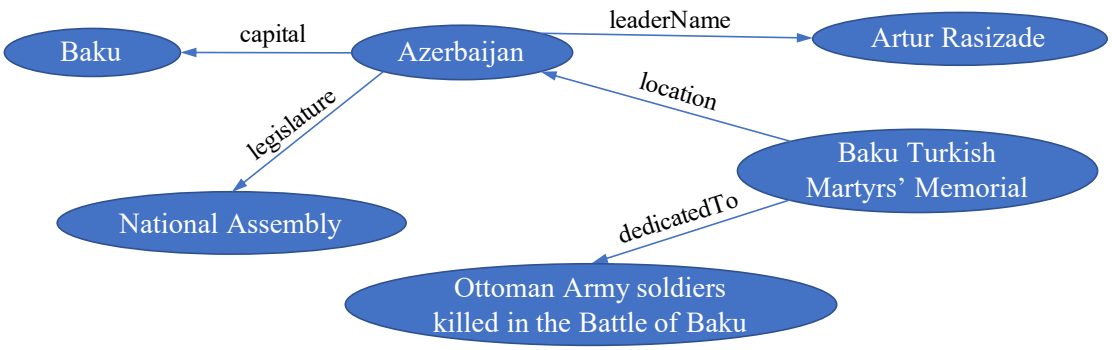
Table 1: Quality evaluation results on the test sets of both English and Russian datasets. Note that models without declaring decoding strategy use Beam Search. For reference, we also report results from previous work in the English dataset. Boldface shows the best result for a column, and arrows indicate the direction of improvement, i.e., ↑: higher is better.

	English			Russian		
	Self-B↓	**MS-B↑**	**MS-C↑**	**Self-B↓**	**MS-B↑**	**MS-C↑**
Deterministic Models						
Top-3 beam search	86.72	46.65	71.45	76.50	38.23	61.58
Total random sampling	**56.48**	40.47	67.00	**52.30**	31.37	56.30
Top-3 random sampling	64.66	45.15	70.40	60.31	35.61	59.95
MC dropout	68.70	46.90	70.87	61.59	36.14	59.37
Ensemble	81.31	47.32	71.52	75.70	38.50	61.71
Stochastic Models						
Single model	97.30	43.25	69.45	97.62	33.53	58.40
Ensemble	77.85	**47.61**	**71.95**	73.50	**38.86**	**61.95**

Table 2: Diversity evaluation results on the test sets of both English and Russian datasets. **Self-B** refers to Self-BLEU while **MS-B** and **MS-C** refer to the proposed Multi-Score metric using sentence-level BLEU and CHRF++ as the underlying quality metric. Note that models without declaring decoding strategy use beam search decoding.

Models	**Syntax/Spelling Mistake**	**Lack of Information**	**Information Redundancy**	**Average**
Deterministic Models				
Total random sampling	0.54	0.18	0.20	0.33
Top-3 random sampling	0.18	0.14	0.49	0.22
MC dropout	0.18	0.32	0.20	0.22
Top-3 beam search	0.07	0.14	0.00	0.09
Ensemble	0.00	0.09	0.03	0.06
Stochastic Models				
Ensemble	0.03	0.13	0.08	0.08

Table 3: Error analysis results, showing the rates of mistakes for each model.



DM (MC dropout) 1: The Baku Turkish Martyrs' Memorial, which is dedicated to the Ottoman Army soldiers killed in the battle of Baku, is found in Azerbaijan. The capital of Azerbaijan is Baku and the leader is Artur Rasizade. (missing: legislature information)
DM (MC dropout) 2: The Baku Turkish Martyrs' Memorial, which is dedicated to the Ottoman Army soldiers killed in the battle of Baku, is dedicated to the Ottoman Army soldiers killed in the country is led by Artur Rasizade. (missing: legislature information)
DM (MC dropout) 3: The Baku Turkish Martyrs' Memorial is dedicated to the Ottoman Army soldiers killed in the battle of Baku. It is dedicated to the Ottoman Army soldiers killed in the battle of Baku, the leader of Azerbaijan is Artur Rasizade. (missing: legislature information)

SM (Ensemble) 1: The Baku Turkish Martyrs' Memorial is dedicated to the Ottoman Army soldiers killed in the battle of Baku. It **is located in** Azerbaijan **whose capital is** Baku and its leader is Artur Rasizade. **The legislature is** the National Assembly.
SM (Ensemble) 2: Baku **is the capital of** Azerbaijan **where the legislature is** the National Assembly and the leader is Artur Rasizade. The country **is the location of** the Baku Turkish Martyrs Memorial **which is dedicated to** the Ottoman Army soldiers killed in the battle of Baku.
SM (Ensemble) 3: The Baku Turkish Martyrs' Memorial **is dedicated to** the Ottoman Army soldiers killed in the battle of Baku. It is located in Azerbaijan whose capital is Baku and its leader is Artur Rasizade**, and its legislature is** the National Assembly.

Table 4: A WebNLG input graph and the outputs from a Deterministic Model (MC dropout) and a Stochastic Model (Ensemble). Highlighted segments indicate mistakes: red, dotted lines represent Syntax/Spelling mistakes, blue, solid lines corresponds to Lack of Information, and orange, dashed lines represent Information Redundancy. **Bold** segments show examples of syntactic variations.

lack of information still remains a challenge for some instances in this setting. Addressing this problem is an avenue that we leave for future work.

6.2 Case Study

Table 4 shows an instance of a semantic graph from which we collect three outputs from a deterministic model (MC dropout) and a stochastic model (Ensemble). The outputs from MC dropout contain three types of mistakes and have low diversity. While there is no mistake in the outputs of the stochastic model, and the boldface illustrates syntactic variation.

7 Conclusion and Future Work

In this work, we first propose stochastic graph-to-text models to generate diverse sentences from semantic graphs. This was implemented through latent variable models that aim to capture linguistic variation and ensembling techniques. Furthermore, to solve the limitation of the existing diversity evaluation metrics, we also propose Multi-Score, a new automatic evaluation metric assessing diversity and quality jointly. It provides a general and effective way to assess the diversity of generated sentences for any text generation task. We perform experiments on English and Russian datasets and results demonstrate the generated sentences from the stochastic ensemble have both high diversity and high quality.

Since Multi-Score is based on using the reference set as the gold standard, it has a limitation that the variety of the reference sentences can largely influence the metric. Datasets containing reference sentences with higher quality and diversity will likely generate a more accurate Multi-Score for the predicted sentences. In other words, Multi-Score evaluates diversity *implicitly* through the references, as opposed to *explicit* judgements of diversity. However, explicit human evaluation requires a formal definition of diversity which is difficult to establish (as compared to quality judgements, for

instance). Nevertheless, addressing this challenge could provide a pathway to reduce the need for multiple references in evaluating diversity.

To the best of our knowledge this is the first work that incorporates a latent variable within a graph-to-sequence model. This in turn leads to many promising research avenues to explore in future work. Our analysis showed that the latent variable mostly helps in syntactic variation but less in other aspects such as semantics. Analysing the behaviour of the latent variable when modelling linguistic information is an important avenue that will enhance the understanding of stochastic models.

References

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 273–283. Association for Computational Linguistics.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 10–21. ACL.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 552–562. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1050–1059. JMLR.org.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 179–188. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. The webnlg challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133. Association for Computational Linguistics.

Frank van Harmelen, Vladimir Lifschitz, and Bruce W. Porter, editors. 2008. *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*. Elsevier.

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

Daphne Ippolito, Reno Kriz, João Sedoc, Maria Kustikova, and Chris Callison-Burch. 2019. Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3752–3762. Association for Computational Linguistics.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2284–2293. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR:

sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 146–157. Association for Computational Linguistics.

Harold W. Kuhn. 2010. The hungarian method for the assignment problem. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 29–47. Springer.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6402–6413.

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

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated graph sequence neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

James Lucas, George Tucker, Roger B. Grosse, and Mohammad Norouzi. 2019. Understanding posterior collapse in generative latent variable models. In *Deep Generative Models for Highly Structured Data, ICLR 2019 Workshop, New Orleans, Louisiana, United States, May 6, 2019*. OpenReview.net.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation, Tilburg University, The Netherlands, November 5-8, 2018*, pages 1–9. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290. ACL.

Artidoro Pagnoni, Kevin Liu, and Shangyan Li. 2018. Conditional variational autoencoder for neural machine translation. *CoRR*, abs/1812.04405.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Maja Popovic. 2017. chrf++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, pages 612–618. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. *Trans. Assoc. Comput. Linguistics*, 8:589–604.

Philip Schulz, Wilker Aziz, and Trevor Cohn. 2018. A stochastic decoder for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1243–1252. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. GTR-LSTM: A triple encoder for sentence generation from RDF data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1627–1637. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *CoRR*, abs/1710.10903.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2773–2781.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws,

Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *CoRR*, abs/1804.00823.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2972–2978. IJCAI/AAAI Press.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 521–530. The Association for Computational Linguistics.

Chao Zhao, Marilyn A. Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2481–2491. Association for Computational Linguistics.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1097–1100. ACM.

Neural Methodius Revisited: Do Discourse Relations Help with Pre-Trained Models Too?

Aleksandre Maskharashvili, Xintong Li, Symon Jory Stevens-Guille and Michael White

Department of Linguistics
The Ohio State University
maskharashvili.1@osu.edu znculee@gmail.com
stevensguille.1@osu.edu mwhite@ling.osu.edu

Abstract

Recent developments in natural language generation (NLG) have bolstered arguments in favor of re-introducing explicit coding of discourse relations in the input to neural models. In the Methodius corpus, a meaning representation (MR) is hierarchically structured and includes discourse relations. Meanwhile pre-trained language models have been shown to implicitly encode rich linguistic knowledge which provides an excellent resource for NLG. By virtue of synthesizing these lines of research, we conduct extensive experiments on the benefits of using pre-trained models and discourse relation information in MRs, focusing on the improvement of discourse coherence and correctness. We redesign the Methodius corpus; we also construct another Methodius corpus in which MRs are not hierarchically structured but flat. We report experiments on different versions of the corpora, which probe when, where, and how pre-trained models benefit from MRs with discourse relation information in them. We conclude that discourse relations significantly improve NLG when data is limited.

1 Introduction

The success of neural methods in numerous subfields of NLP lead to recent development of neural 'end-to-end' (e2e) architectures in natural language generation (NLG) (Dušek et al., 2020), where a direct mapping from meaning representations (MRs) to text is learned. While recent neural approaches mostly map flat inputs to texts without representing discourse level information explicitly within MRs, Balakrishnan et al. (2019) argues that discourse relations should be reintroduced into neural generation, echoing what has been long argued in more traditional approaches to natural language processing where discourse relations play one of the central roles in natural language text understanding and generation (Mann and Thompson, 1988; Reiter and Dale, 2000; Lascarides and Asher, 2007).

To study whether discourse relations are beneficial for neural NLG, Stevens-Guille et al. (2020) proposed the Methodius corpus, which was developed as an experiment in recreating the classic rule-based NLG system Methodius (Isard, 2016) using a neural generator. In their corpus, the meaning representation (MR) of a text is a tree that encodes the overall discourse structure of the texts plus facts related by discourse relations therein. They were concerned with whether explicit encoding of discourse relations improves the quality of generated texts by LSTM recurrent neural networks (Hochreiter and Schmidhuber, 1997). However, they left open the question whether discourse relations are helpful for pre-trained transformer-based (Vaswani et al., 2017) language models (Lewis et al., 2020; Raffel et al., 2019), which have recently shown remarkable performance on NLG tasks. In this work, we address that question using the T5-Large implementation of Wolf et al. (2019).

A particularly attractive quality of pre-trained models is their ability to generalize from limited data. For example, Peng et al. (2020) proposed to fine-tune a model pre-trained on a large NLG corpus using a small amount of labeled data from a specific domain to adapt the model to generate texts in that domain. In a similar vein, when the labeled data is limited, Arun et al. (2020) suggest to use a large pre-trained model with self-training and knowledge distillation to smaller, faster models. Kale and Rastogi (2020) argue that pre-trained language models make it possible to transform a sequence of semantically correct, but (possibly) ungrammatical template-based texts into a natural sounding, felicitous text of English. They find that template-based textual input is beneficial to use with pre-trained language models when the model needs to generalize from relatively few examples.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 12–23,
Aberdeen, Scotland, UK, 20-24 September 2021.

Given these considerations, we cannot answer the question whether it is helpful to include discourse relations in the input to a pre-trained model for NLG without considering the form of the input, the size of the training data, and the extent to which the test data goes beyond what has been seen in training. As such, we conduct experiments using several versions of the Methodius corpus, where these versions possess one or more of the following properties: (a) discourse relations included in the MR; (b) discourse relations excluded from the MR; (c) tree-structured MR (a hierarchically structured representation of the meaning); (d) flat, textual MR (i.e., non hierarchically structured). We are furthermore concerned with how the linguistic knowledge encoded in pre-trained language models interacts with the different versions of the corpus. We want to be able to scrutinize the structure of the outputs, i.e., texts, too since our intention is to check the models' capabilities in realizing particular phenomena. For these purposes, we conduct experiments using the following setup: (1) Use various portions of the labeled data. (2) Train zero-shot models (with respect to certain discourse-related phenomena) together with various few-shot models (with respect to the same phenomena). (3) Test various aspects of generated texts, both with respect to discourse structure congruence and correctness (factual information).

2 Re-lexicalized & Flat Versions of Methodius

The Methodius system (Isard, 2016) uses discrete rules to generate texts containing predefined subtexts, such as descriptions of exhibits and historical facts about certain periods. To avoid data sparsity and long sequences, Stevens-Guille et al. (2020) delexicalize texts as they substitute certain parts of text by tokens, which they dub special terminals. We want to take advantage of pre-trained language models, which are not exposed to these tokens. Tokens should be substituted by text wherever possible to ensure the input is consistent with the texts the pre-trained models were trained on. However, using some predefined morpho-syntactic constructions and lexical items makes it more manageable to check whether a model performs well with respect to automatic checks. Moreover, if the model experiences problems on such data, it suggests the model would have problems with even less homogeneous data.

Instead of training the models directly on the Methodius corpus or texts harvested through crowdsourcing, we modify the Methodius corpus (i.e., MRs paired with texts) by substituting custom homogeneous texts for the Methodius corpus's special terminals. We substitute some predetermined names for named entities in the Methodius corpus to further homogenize the inputs. This procedure deterministically rewrites the texts in the corpus of Stevens-Guille et al. (2020) into pure English texts and thus maintains the homogeneity of the Methodius corpus. The corresponding MRs are also rewritten into their lexicalized versions.[1] Moreover, we transform Rhetorical structure theory (Mann and Thompson, 1988) style hierarchically structured meaning representations of Methodius texts into a flat, textual input by translating every fact and every discourse relation into a sequence of sentences. Figure 1 shows an MR from the Methodius corpus, the corresponding text from the Methodius corpus, and the new MR that we have substituted for the Methodius corpus MR.

3 Models: RSTSTRUCT, FACTSTRUCT, RSTT2T, and FACTT2T

We fine-tune T5-large (Raffel et al., 2019) on the following types of labeled data:

- Input MRs from the Methodius corpus modified by the procedure described in the foregoing (see Figure 1b). It contains discourse relations. We dub the result RSTSTRUCT.
- Input MRs obtained by erasing discourse information from the inputs of RSTSTRUCT. This amounts to deleting discourse relation markers (SIMILARITY and CONTRAST) in the inputs of RSTSTRUCT. We dub the result FACTSTRUCT.
- Input MRs obtained by transforming the MRs of RSTSTRUCT into flat, purely textual representations (see Figure 1c).[2] We dub the result RSTT2T.
- Input MRs obtained by removing discourse information from RSTT2T MRs. This amounts to deleting discourse relation markers ('however' and 'likewise') in the inputs of RSTT2T. We dub the result FACTT2T.

[1] The code can be found at https://github.com/aleksadre/methodiusNeuralINLG2021.

[2] We have defined a set of rules that transform hierarchically structured MRs into texts.

Figure 1: A Methodius MR, the re-lexicalized MR, and its flat, textual version, together with the surface realization

(a) Delexicalized meaning representation from Methodius corpus

```
[__content_plan
 [__rst_elaboration
  [__fact_type [__arg1 entity0 ] [__arg2 statue ] ]
  [__rst_joint [__fact_made_of [__arg1 entity0 ] [__arg2 material_0 ] ]
               [__fact_exhibit_portrays [__arg1 entity0 ] [__arg2 god_0 ] ] ] ]
 [__rst_contrast
  [__fact_creation_period compare_additive [__arg1 entity1 ]
                                           [__arg2 historical_period_0 ] ]
  [__fact_creation_period [__arg1 entity0 ] [__arg2 historical_period_1 ] ] ]
 [__optional_type [__arg1 entity1 ] [__arg2 vessel ] ] ]
```

(b) Lexicalized meaning representation of the foregoing (we treat tokens of the form '$[\_\_xyz$' and '$]$' as indivisible tokens in our experiments)

```
[__content_plan
 [__rst_elaboration
  [__fact_type [__arg1 entity0 ] [__arg2 statue ] ]
  [__rst_joint [__fact_made_of [__arg1 entity0 ] [__arg2 bronze ] ]
               [__fact_exhibit_portrays [__arg1 entity0 ] [__arg2 apollo ] ] ] ]
 [__rst_contrast
  [__fact_creation_period compare_additive [__arg1 entity1 ]
                                           [__arg2 classical period ] ]
  [__fact_creation_period [__arg1 entity0 ] [__arg2 hellenistic period ] ] ]
 [__optional_type [__arg1 entity1 ] [__arg2 vessel ] ] ]
```

(c) Flat, textual meaning representation

```
this statue is a statue. this statue is made of bronze. this statue portrays apollo.
the previously seen vessel was created in the classical period.
however this statue was created in the hellenistic period.
```

Text: This is a statue; it is made of bronze and it portrays Apollo. Unlike the vessel you recently saw, which was created during the classical period, this statue was created during the hellenistic period.

Figure 2: Instances of constructions starting with SIMILARITY and CONTRAST, which are not included in zero-shot data

(a) The Like Construction and the corresponding text

```
[__content_plan
    [__rst_similarity
        [__fact_original_location [__arg1 entity1 ] [__arg2 attica ] ]
        [__fact_original_location [__arg1 entity0 ] [__arg2 attica ] ] ]
    [__fact_exhibit_story [__arg1 entity0 ] [__arg2 it was part of a collection dedicated to athena ] ]
    [__fact_current_location [__arg1 entity0 ] [__arg2 the national archaeological museum ] ]
    [__fact_exhibit_depicts [__arg1 entity0 ] [__arg2 the goddess athena ] ]
    [__optional_type [__arg1 entity0 ] [__arg2 lekythos ] ]
    [__optional_type [__arg1 entity1 ] [__arg2 kylix ] ] ]
```

Text: Like the kylix you recently saw, this lekythos originates from Attica. It was part of a collection dedicated to Athena. This lekythos is located in The National Archaeological Museum. It depicts the goddess Athena.

(b) The Unlike Construction and the corresponding text

```
[__content_plan
    [__rst_contrast [__fact_original_location [__arg1 entityplural ] [__arg2 attica ] ]
    [__fact_original_location [__arg1 entity0 ] [__arg2 macedonia ] ] ]
[__fact_exhibit_story [__arg1 entity0 ] [__arg2 it was part of a collection dedicated to athena ] ]
    [__fact_current_location [__arg1 entity0 ] [__arg2 the national archaeological museum ] ]
    [__fact_exhibit_depicts [__arg1 entity0 ] [__arg2 the goddess athena ] ]
    [__optional_type [__arg1 entityplural ] [__arg2 vessel ] ]
    [__optional_type [__arg1 entity0 ] [__arg2 tetradrachm ] ] ]
```

Text: Unlike the vessels you recently saw, which were originally from Attica, this tetradrachm originates from Macedonia. It was part of a collection dedicated to Athena. Now this tetradrachm is exhibited in The National Archaeological Museum. It shows the goddess Athena.

We refer to models by the name of the data type they are fine-tuned on.

Name	Size 100%	Tok. Av.	SIM.	CONTRA.
Training	4222	180	2892	777
Validation	417	181	290	76
Challenge Test	237	96	80	80
Standard Test	799	134	495	166

Table 1: Size of training, validation and test sets; average tokens per pair (MR,text); numbers of SIMILARITY and CONTRAST relations in data sets.

In addition to using the whole dataset for training, we conduct experiments on (randomly selected) 1%, 3%, 5%, 10%, 20%, and 50% portions of the data. With 100% percent data, we train each model three times, while for the sub portions of the data set, we train the models five times each (each time we select random dataset of that portion). This lets us get an idea of the variance between different runs of the same model.

We distinguish three further subtypes of data, calling them zero-, few- and ten-shot data (which we also denote by prefixes Z-, F-, and D-, respectively). In zero-shot data, none of the MRs beginning with SIMILARITY or CONTRAST, the surface realization of which would start with 'Like' or 'Unlike,' are included in the training data. These constructions are exemplified in Figure 2. When constructing the few-shot data, the foregoing restriction on the form of the MRs is removed. But in each portion of the few-shot training data, we include only three examples of each construction. When constructing the ten-shot data, ten instances of each of the constructions that were introduced in the few-shot data are included. Tying the number of these constructions to the size of the dataset lets us more effectively compare a model behavior with and without these constructions.

4 Evaluation Methods

We adopt the double test set style from Stevens-Guille et al. (2020). One test set is called Standard and the other is called Challenge (see Table 1). There are several differences between them. The Standard test set examples are independently selected, while the Challenge test set examples are not. In the Challenge test, around 12% of the test items have structure not observed in the training set for zero-shot models.

4.1 Types of Errors

Discourse Relation Errors We are interested in observing the performance of the models with respect to generating coherent discourse relations. While there are several discourse relations in the Methodius corpus, we focus on CONTRAST and SIMILARITY for several reasons. First, they are interesting in terms of their meaning—they require identifying whether properties or entities are co-extensive or distinct, but can be inferred from the facts alone. Second, there is a consistent method of expressing them in the Methodius corpus outputs: SIMILARITY is realized by *like* and CONTRAST is realized by *unlike*.

Repetitions, Hallucinations, and Omissions (RHOs) Given the way the revised Methodius corpus is constructed, its texts follow certain predetermined lexical and morpho-syntactic patterns. We use this property of the texts to measure the performance of models with respect to the following errors: hallucination of content; omission of content; and repetitions of content. To be more precise, for every test item we compare the model output and the reference text by determining their difference with respect to the *special terminals*, i.e., the content that is obtained by relexicalizing the delexicalized content).

Lexical Hallucinations Since Methodius is designed purposely to be homogeneous, it is useful to measure how many novel strings pre-trained models come up with when fine-tuned on data that does not contain these strings. For that, we count per test set the lexical hallucinations, i.e., items produced by the model which are not observed in the corpus.

Mistaken Role Identity (mistID) We sometimes observe a mismatch between the exhibit type in the input and its realization in the output. For instance, in Example (1), 'imperial portrait' and 'vessel' are swapped, i.e, their roles are misidentified. We consider this kind of error distinct from the previous error types and refer to it by mistID.

(1) Ref: This is a vessel; it was created between 500 and 480 B.C. Unlike the imperial portraits you recently saw, which were originally from Attica, this vessel was originally from Acropolis.
Gen: This imperial portrait was created between 500 and 480 B.C. Unlike the vessels you recently saw, which originate from Attica, this imperial portrait was originally from Acropolis.

4.2 Statistical Significance: Stratified Approximate Randomization

To compare various models, we use stratified approximate randomization (AR; Noreen 1989), which is a powerful and generic method of establishing significant differences between models. One advantage of AR over more traditional paired tests for NLP tasks is that it does not require independence of samples, which is usually violated when we consider various runs of the same model on the same test set (as the same test item gets tested several times by the same model) (Clark et al., 2011). In the present work, we rely on stratified AR to identify whether differences between the performance of various models over several runs is significant. (The description of the stratified AR algorithm is provided in Section A.1 of Appendix A.)

5 Results

Below, we report results on the data portions 1%, 3%, 5%, 10%, 20%, and 50% of the few-shot models (results on the corresponding zero-shot data models are provided in Appendix A.4). For 100% data usage, we report results of zero-, few-, and ten-shot models.

5.1 Data portions: 1%, 3%, and 5%

Discourse Relations: As Figure 3 indicates, there are fewer errors in discourse relation realization for T2T (RSTT2T and FACTT2T) models compared to structure models (RSTSTRUCT and FACTSTRUCT).

RHOs: Figure 4 shows the number of RHO mistakes the models make. T2T models make less RHO mistakes compared to structure models. Also, each of the models produces a large number of lexical hallucinations, but T2T are less prone to lexical hallucinations compared to structured models as RSTSTRUCT and FACTSTRUCT each produce on average 100 lexical hallucinations at 1% and 50 lexical hallucinations at 3% and 5% data portion, whereas T2T make only third of those lexical hallucinations on each of the data portion (for full details on various runs see Figure 9 in Appendix A). We note that at the 1% portion of the data, the quality of generated texts is unsatisfactory, even by T2T models. This can be seen by automatic metrics, as well as by eyeballing the generated texts. On 3% and 5%, the quality gets slightly better for structured models and we see more rapid improvements for T2T models.

Summary: RST vs. FACT The question whether models with discourse relations (RSTSTRUCT and RSTT2T) perform better than ones without discourse relations (FACTSTRUCT and FACTT2T respectively) can be answered positively. As for T2T models, we declare with high confidence that RSTT2T outperforms FACTT2T in every collected statistics. We are not however able to say that for structured models, even though in more than half of the comparisons RSTSTRUCT is at least as good as FACTSTRUCT.

5.2 Data portions: 10%, 20%, and 50%

By using data portions 10%, 20%, and 50%, we see many improvements in quality of texts compared to 1%, 3%, and 5%. Also in this case (i.e. on the data 10%, 20%, and 50%), T2T models show better performance compared to structure models.

Discourse Relations: Figure 5 illustrates that RSTT2T does better or at least as good as FACTT2T. The same can be said about RSTSTRUCT and FACTSTRUCT, with the only exception of the case of the 10% data on the Challenge set as FACTSTRUCT shows better results than RSTSTRUCT.

RHOs: In terms of RHO errors, RSTT2T together with FACTT2T are winners on either test sets, as it is indicated by the results on Figure 6. In addition, by measuring lexical hallucinations, we conclude that the both RSTT2T and FACTT2T are the least hallucinating models (the detailed statistics is given on Figure 9 in Appendix A).

Summary: RST vs. FACT Again, RSTT2T comes out as the winner among all models (vs. RSTSTRUCT, FACTSTRUCT, and FACTT2T) by all the evaluation metrics involved. It must be noted though that as we reach 50%, we do not see significant differences between RSTT2T and FACTT2T. Also, RSTSTRUCT does better or at least as good as FACTSTRUCT, except for one case.

5.3 100%: Zero, Few, and Ten shot

In 100% data, we see less difference among performance of models as structured models show visible improvement, catching up with T2T models. Below, we compare models trained on zero-, few- and ten-shot data.

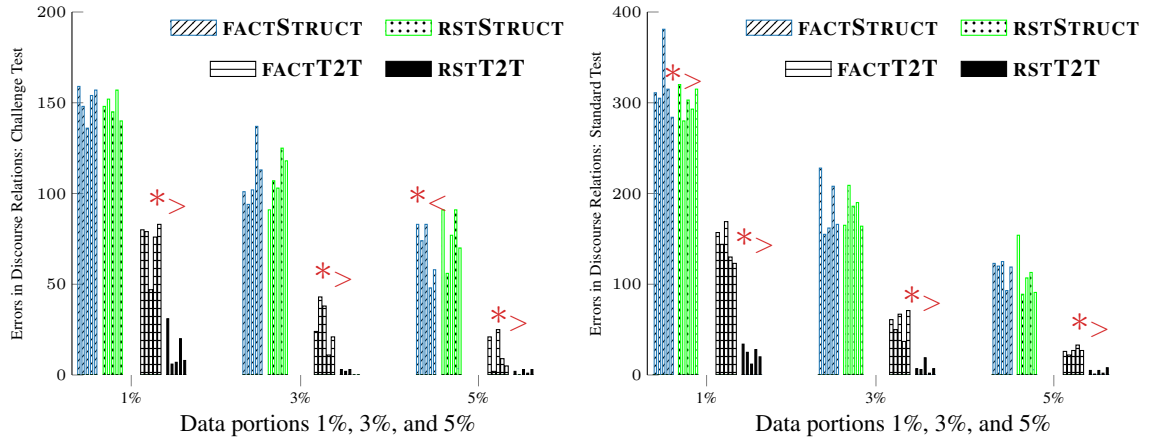


Figure 3: Few Shot Models: Discourse relation realization on the Challenge and Standard tests (A $*_>$ B or B $*_<$ A indicate that the model A has significantly more errors than the model B, where the significance level is set to 0.05; we use this convention in all figures)

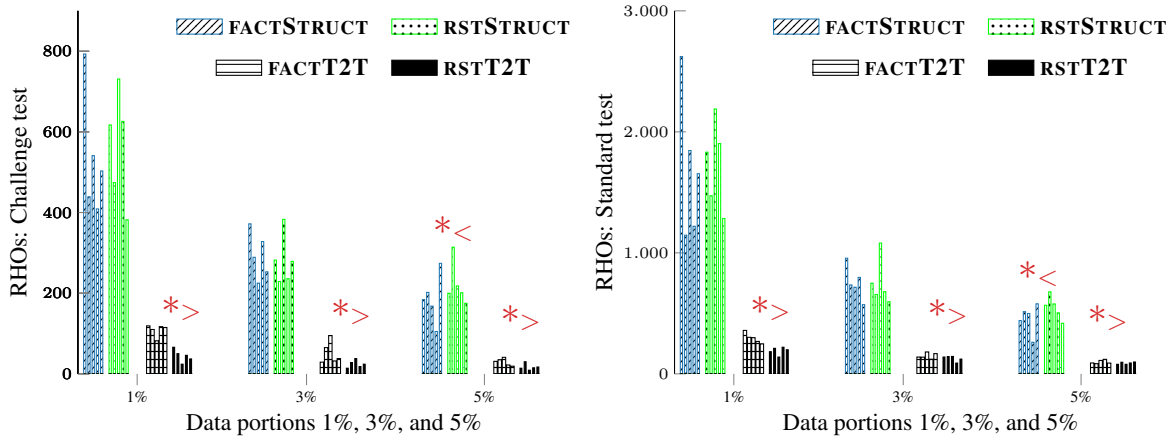


Figure 4: Few Shot Models: RHOs on the Challenge and Standard tests

Discourse Relations: Few-shot models realize discourse connectives on average better than zero-shot models, even when deploying 100% of the training data, which we see in Figure 2. On the Challenge set, which contains the constructions whose lookalikes are not contained at all in the zero-shot training data, it is not unexpected that few-shot models perform better. But, even on the Standard test set, we see that few-shot models are better than zero-shot. That being said, we can see that on one of the runs a zero-shot model achieves one of the best scores. We can say that few-shot models are more consistent than zero-shot models; moreover, they are beneficial when training models on small data sets where models may not have enough data to generalize over every possible phenomenon.

RHOs: In Figure 8, we see that RHOs are lower than in cases of 50% data usage. But nevertheless, they are present. Here as well, the best performing models are T2T models. On the Challenge set, few-shot and ten-shot models make less mistakes than zero-shot models. This indeed is correlated with the fact that few-shot and ten-shot models perform better in terms of discourse relations: By realizing discourse structure correctly, the model needs to repeat or omit less information than by making a mistake and then either repeating the same information again or omitting it because it does not fit into the structure it has been building.

We also found that 100% models make very few lexical hallucinations (usually 0). However, we see mistID errors in 100% models almost as many as

Model Name	Z-100	F-100	D-100	F-50	F-5
FACTSTRUCT	2	2	1	4	12
RSTSTRUCT	**4**	**10**	**3**	**8**	**8**
FACTT2T	3	0	0	7	0
RSTT2T	3	0	0	0	3

Table 2: Maximum of mistID errors of models on the Challenge test set

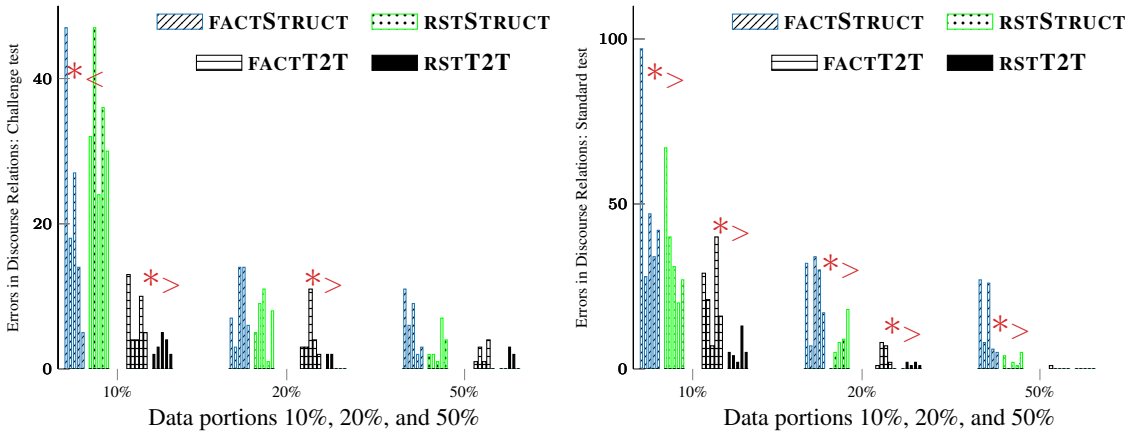


Figure 5: Few Shot Models: Discourse relation realization on the Challenge and Standard tests

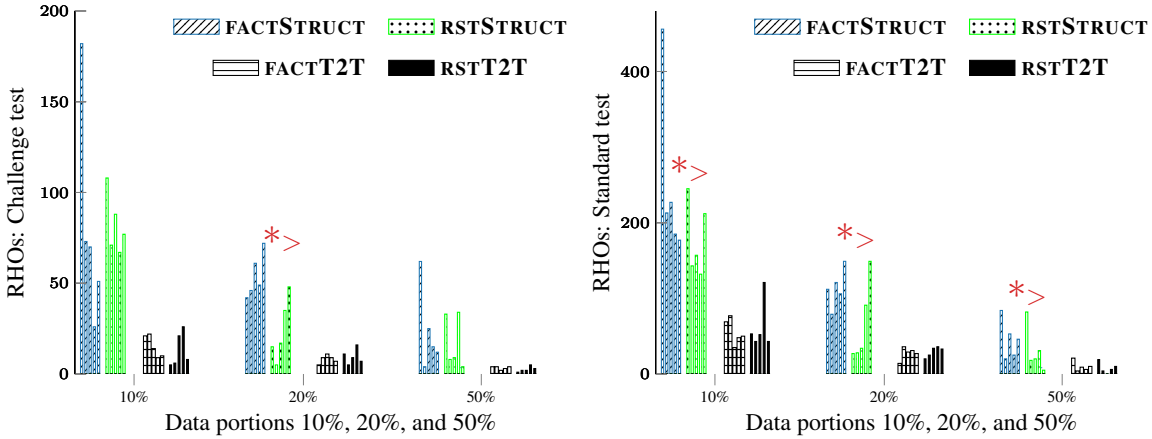


Figure 6: Few Shot Models: RHOs on the Challenge and Standard tests

we see in models trained on small portions of data. Table 2 shows that RSTSTRUCT models trained on Z-100 and F-100 data do not have significant improvements over the models trained on smaller portions of data. It may also seem that a zero-shot RSTSTRUCT model does better than a few-shot one. We have closely examined those cases. The mistID errors arise in those cases whose lookalikes have not been seen by zero-shot models, i.e., the ones similar to the cases shown in Figure 2b and Figure 2a. Zero-shot models either skip some of the comparisons or do it differently—as shown in Example (2), Z.a makes the comparison differently from Ref, whereas Z.b skips it entirely. This is apparently why zero-shot models do not produce as many mistID errors. By contrast, few-shot models are able to recognize those constructions (as they have seen three of each in training) and try to realize them, which they do quite successfully but in so doing they may commit mistID errors, as shown in Example (2), F.

(2) Ref: Like the kylix you recently saw, this lekythos originates from Attica. It was part of a collection dedicated to Athena. Now this lekythos is exhibited in The National Archaeological Museum. It shows the goddess Athena.

F: Like the lekythos you recently saw, this kylix was originally from Attica. It was part of a collection dedicated to Athena. Now this kylix is exhibited in The National Archaeological Museum. It shows the goddess Athena.

Z.a: This lekythos originates from Attica. Like the kylix, this lekythos was originally from Attica. It was part of a collection dedicated to Athena. This lekythos is currently in The National Archaeological Museum. It depicts the goddess Athena.

Z.b: This is a lekythos and it was originally from Attica. It was part of a collection dedicated to Athena. This lekythos is currently in The National Archaeological Museum. It depicts the goddess Athena.

This hypothesis can be checked by looking at ten-shot model performance: They have fewer mistID errors than few-shot models, presumably because they are more comfortable with those constructions, as they see them more than few-shot models. If we look at F-50% models (see Table 2), they do not do worse on mistIDs than F-100% models. That

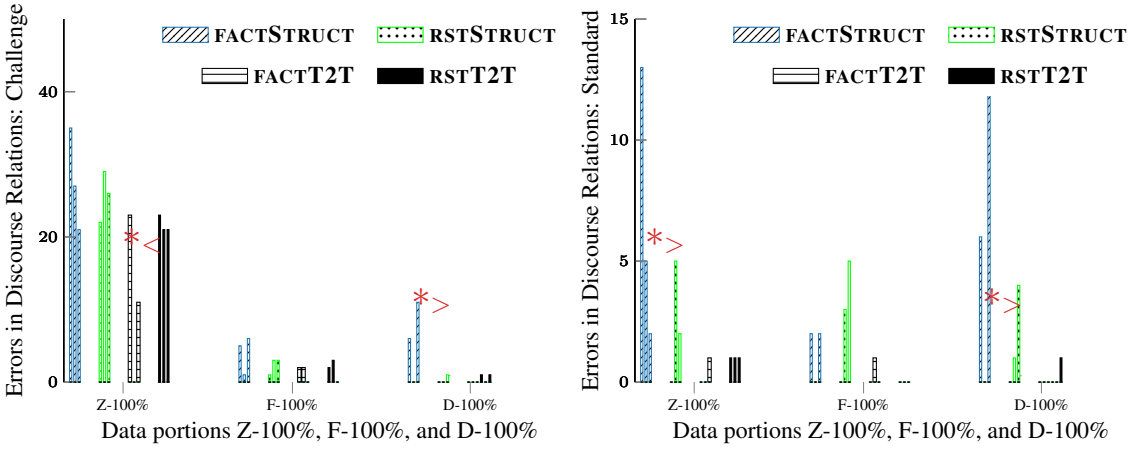


Figure 7: 100% models, Zero-, Few-, and 10-Shot: Discourse relation realization on the Challenge and Standard tests

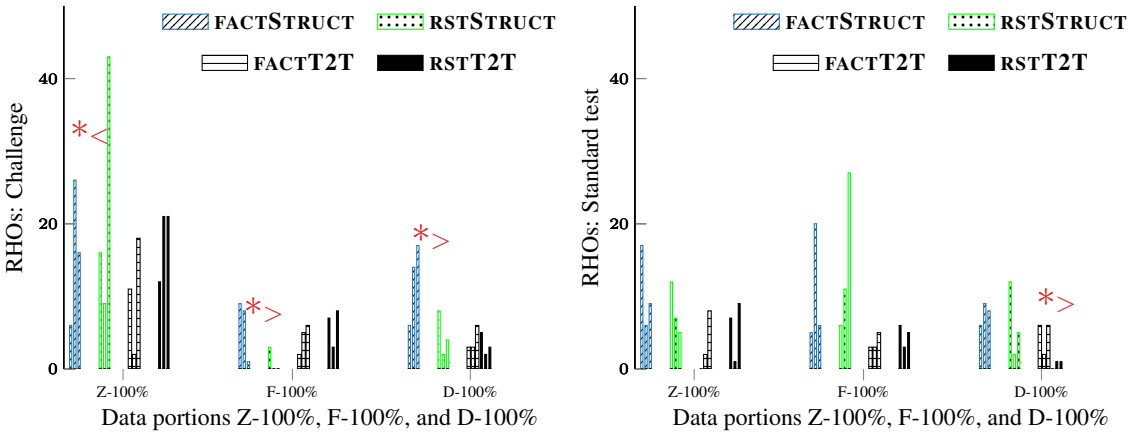


Figure 8: 100% models, Zero-, Few-, and 10-Shot: RHOs on the Challenge and Standard tests

could perhaps be explained by the fact that with F-50% models and D-100% models, both have the same relative number of constructions of interest, which means that 50% models have twice as high concentration of those examples compared to the corresponding 100% few-shot models.

Summary: RST vs. FACT At 100% data, all models show more or less the same performance according to the metrics we use. In terms of hallucinations, we detected that only on Z-100 data, FACTSTRUCT model was prone to hallucinating 'large vessel.' We also found that in D-100 data, FACTSTRUCT has high variance, both in lexical hallucinations and RHOs.

6 Discussion and Conclusion

The development of neural NLG led to an understandable focus on simpler phenomena; the networks in currency at the time seemed to perform best on short, entity-focused texts. While new methods frequently make progress by working on simple domains, we echo the conclusions of Stevens-Guille et al. (2020) that neural methods can and should address more complex, rhetorically structured text, which they must if they are to produce genuinely coherent discourses (Prasad et al., 2008). Our results here bolster those conclusions and provide further evidence for the usefulness of explicit discourse coding in the input to neural systems, especially when data is limited in size. In line with the contemporary wisdom concerning pre-trained models, our results suggest that fine-tuning such models when labeled data for specific domains is limited improves the felicity of generated texts. While increases in available data do always improve the quality of generated texts in terms of grammatically and correctness, we see fast and dramatic improvements when using text inputs, with only more gradual increases in quality in the case of structured input. But we stress that dis-

course relations are enormously helpful when the dataset for the domain is limited: at lower levels of data usage, RSTT2T consistently significantly outperforms FACTT2T on every metric we use. Given the benefits of explicitly encoding discourse relations in the input to the models reported here, we conclude by recommending the continued development of NLG corpora in which discourse relations are present in the meaning representations.

For today, even though various corpora have been designed for natural language generation purposes, corpora with discourse structure information are not available. Given our results showing the benefits of having discourse information in the input, we hope that more corpora will be designed where discourse information is provided with the help of discourse relations.

Acknowledgments

We thank Amy Isard for helping us with Methodius. We are thankful to three anonymous reviewers for their helpful comments. We also want to thank The Ohio Super Computer Center (Center, 1987) for their support as they provided us with needed computational power. This research was supported by a collaborative open science research agreement between Facebook and The Ohio State University. The last author is a paid consultant for Facebook.

References

Ankit Arun, Soumya Batra, Vikas Bhardwaj, Ashwini Challa, Pinar Donmez, Peyman Heidari, Hakan Inan, Shashank Jain, Anuj Kumar, Shawn Mei, Karthik Mohan, and Michael White. 2020. Best practices for data-efficient modeling in NLG:how to train production-ready neural models with less data. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 64–77, Online. International Committee on Computational Linguistics.

Anusha Balakrishnan, Vera Demberg, Chandra Khatri, Abhinav Rastogi, Donia Scott, Marilyn Walker, and Michael White. 2019. Proceedings of the 1st workshop on discourse structure in neural nlg. In *Proceedings of the 1st Workshop on Discourse Structure in Neural NLG*.

Ohio Supercomputer Center. 1987. Ohio supercomputer center.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language*, 59:123–156.

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

Amy Isard. 2016. The methodius corpus of rhetorical discourse structures and generated texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1732–1736, Portorož, Slovenia. European Language Resources Association (ELRA).

Mihir Kale and Abhinav Rastogi. 2020. Template guided text generation for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6505–6520, Online. Association for Computational Linguistics.

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

Alex Lascarides and Nicholas Asher. 2007. Segmented discourse representation theory: Dynamic semantics with discourse structure. In H. Bunt and R. Muskens, editors, *Computing Meaning: Volume 3*, pages 87–124. Kluwer Academic Publishers.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text & Talk*, 8(3):243 – 281.

Eric W. Noreen. 1989. *Computer-intensive methods for testing hypotheses : an introduction*. Wiley, New York.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Symon Stevens-Guille, Aleksandre Maskharashvili, Amy Isard, Xintong Li, and Michael White. 2020. Neural NLG for methodius: From RST meaning representations to texts. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 306–315, Dublin, Ireland. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

A Appendix 0

A.1 Stratified Approximated Randomization (AR)

The principle behind of Stratified Approximated Randomization (AR) test can be explained as follows: Given that model A output on strata of size $n > 0$ (e.g. a test item can be a stratum) are $a_1 \dots a_n$ and model B outputs on the same n strata are $b_1 \dots b_n$, the performance of the models A and B can be considered significantly different if by swapping a_i with b_i with probability 0.5 would result in a sequence $a_1^{'} \dots a_n^{'}$ (i.e., for every $i \in \{1..n\}$, $a_i^{'}$ is a_i with probability 0.5 or b_i with probability 0.5) and a sequence $b_1^{'} \dots b_n^{'}$ (where, for every $i \in \{1..n\}$, $b_i^{'}$ is a_i with probability 0.5 or b_i with probability 0.5) usually differ less from each other than the original sequences $a_1 \dots a_n$ and $b_1 \dots b_n$ differ from each other.

One may take in the role of a_i (where $i \in \{1..n\}$), not just single output of a model, but a set of outputs obtained by several different runs of the same model. That is, we can have $a_i = \{r_1^1, r_i^2, \dots, r_i^k\}$ where $k \geq 2$ and r_i^l is the output of the l-th run of the model A on the stratum i. Below, we assume that each r_i^l has a numerical value. This allows us to compare two models A and B, each run k times with their respective outputs.

We first compute the expectation (mean) of the sample $a_1 \dots a_n$ by taking mean of each set $a_i = \{r_1^1, r_i^2, \dots, r_i^k\}$ and then calculating their mean. We denote it by m_A. We do the same for the sample $b_1 \dots b_n$ and denote their mean by m_B. Let $d_m = |m_A - m_B|$.

Now we define the following procedure: Construct $a_1^{'} \dots a_n^{'}$ and $b_1^{'} \dots b_n^{'}$ by swapping $a_i = \{r_1^1, r_i^2, \dots, r_i^k\}$ with $b_i = \{r_1^1, f_i^2, \dots, f_i^k\}$. Calculate the mean of $a_1^{'} \dots a_n^{'}$ and the mean of $b_1^{'} \dots b_n^{'}$, denote them by $m_A^{'}$ and $m_B^{'}$ respectively. Compute $d_m^{'} = |m_A^{'} - m_B^{'}|$. We perform this procedure multiple times, say N. If out of N cases, for p-percent (usually p is 5) or less cases we find that $d_m^{'} \geq d_m$, we say that model A and B are significantly different with significance at $p\%$. (Below, in our experiments we take $N = 1000$ and $p = 5$, which is usually considered to be a sufficient margin of significance.)

A.2 Lexical Hallucinations

Figure 9 shows numbers of lexical hallucinations various models make.

A.3 mistID Statistics

Figure 10, Figure 11, and Figure 12 show mistID errors on various runs and models trained on various data portions.

A.4 Zero-shot Results on Discourse Relation Realization

We report performance of the zero-shot models in terms of generating discourse relations relations on Figures 13, Figures 14, Figures 15, and Figures 16.

B Reproducibility Details

We use the pretrained T5-Large HuggingFace transformer model (Wolf et al., 2019). There are total 737683456 trainable parameters in this model. The T5 models are fine-tuned using cross entropy loss without label smoothing. The learning rate is constantly 2×10^{-5} and the batch size is 8 samples. The optimizer is Adam (Kingma and Ba, 2014) where $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, and the weight decay is 0. The best checkpoint is selected by validation with patience of 10 training epochs. For every experiment, the computing infrastructure we used is an NVIDIA V100 GPU and an Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz CPU.

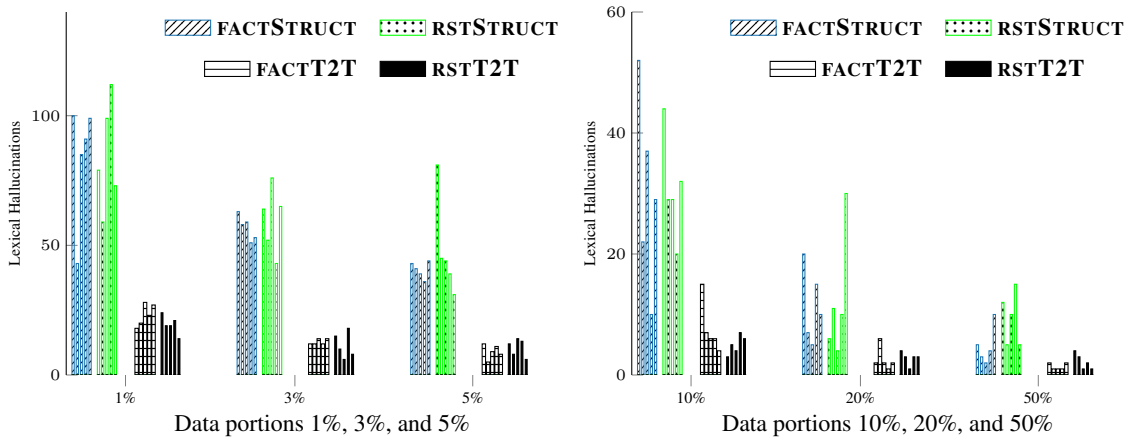


Figure 9: Few Shot Models: Lexical hallucinations combined on the Challenge and Standard tests (no significance tests were performed on lexical hallucinations as they were counted per test set, not per example)

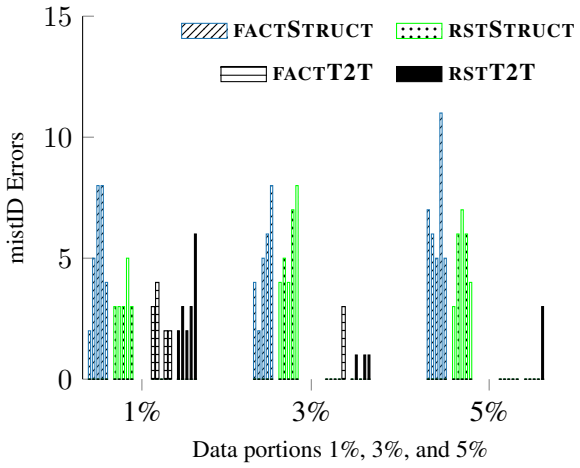


Figure 10: Few Shot Models: mistID errors on the Challange set

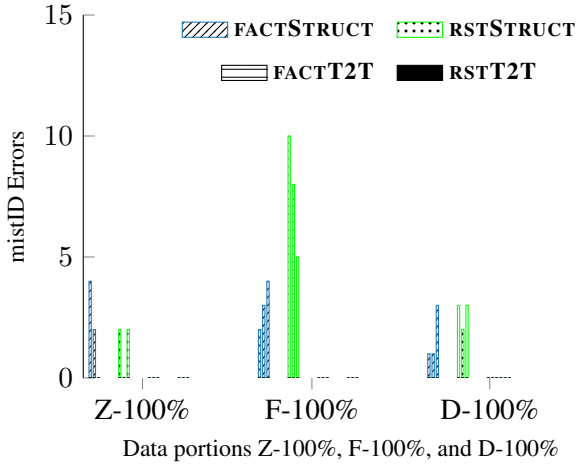


Figure 12: 100% Models: mistID errors on the Challenge test

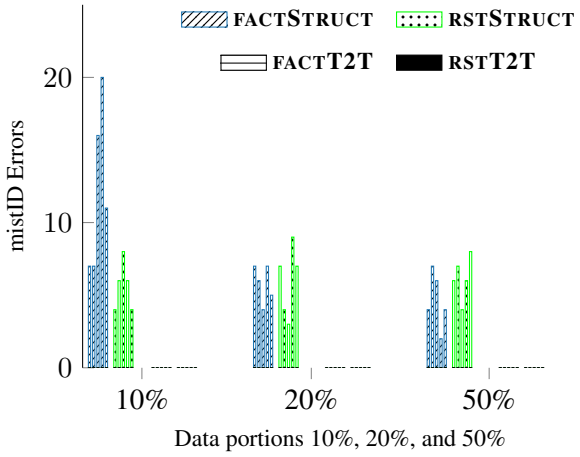


Figure 11: Few Shot Models: mistID errors on the Challenge set

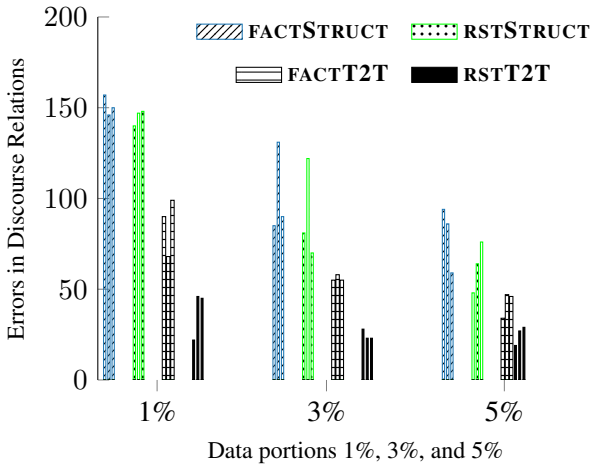


Figure 13: Zero-shot Models: Discourse relation realization on the Challenge test set

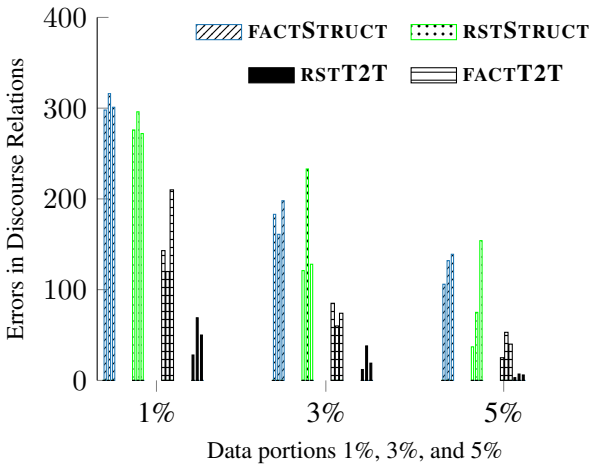


Figure 14: Zero-shot Models: Discourse relation realization on the Standard test set

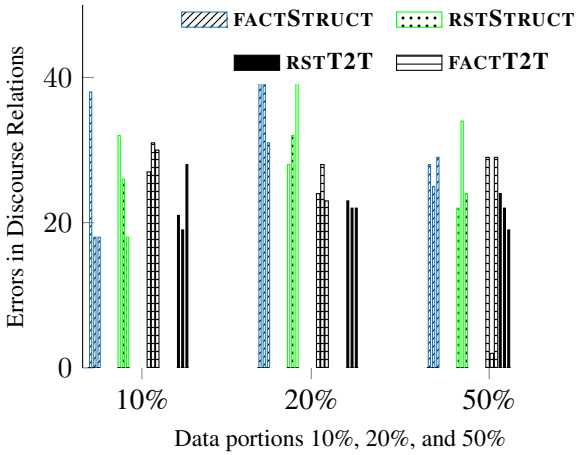


Figure 15: Zero-shot Models: Discourse relation realization on the Challenge test set

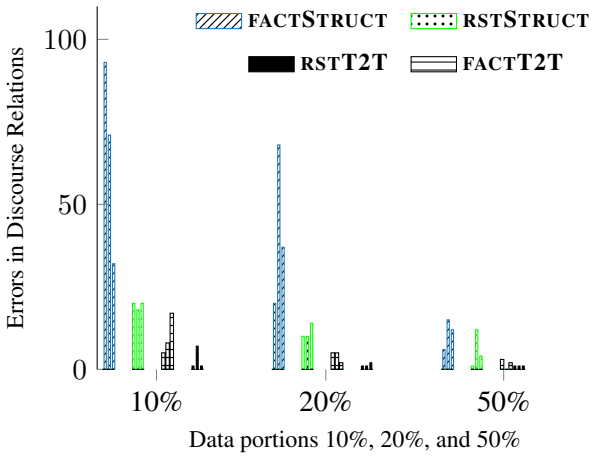


Figure 16: Zero-shot Models: Discourse relation realization on the Standard test set

Exploring Input Representation Granularity for Generating Questions Satisfying Question-Answer Congruence

Madeeswaran Kannan, Haemanth Santhi Ponnusamy,
Kordula De Kuthy, Lukas Stein, Detmar Meurers
University of Tübingen
{mkannan,hsp,kdk,lstein,dm}@sfs.uni-tuebingen.de

Abstract

In question generation, the question produced has to be well-formed and meaningfully related to the answer serving as input. Neural generation methods have predominantly leveraged the distributional semantics of words as representations of meaning and generated questions one word at a time. In this paper, we explore the viability of form-based and more fine-grained encodings such as character or subword representations for question generation.

We start from the typical seq2seq architecture using word embeddings presented by De Kuthy et al. (2020), who generate questions from text so that the answer given in the input text matches not just in meaning but also in form, satisfying question-answer congruence. We show that models trained on character and subword representations substantially outperform the published results based on word embeddings, and they do so with fewer parameters.

Our approach eliminates two important problems of the word-based approach: the encoding of rare or out-of-vocabulary words and the incorrect replacement of words with semantically-related ones. The character-based model substantially improves on the published results, both in terms of BLEU scores and regarding the quality of the generated question. Going beyond the specific task, this result adds to the evidence weighing different form- and meaning-based representations for natural language processing tasks.

1 Introduction

Question generation (QG) is a challenging NLP task, where both language form and meaning play a vital role in the production of questions that have to be well-formed and meaningfully related to the envisaged answer. Neural models have been shown to be very promising for QG, with most recent approaches formulating the task as a sequence learning problem with the goal of mapping a sentence onto a question (e.g., Zhao et al., 2018; Chan and Fan, 2019; Xie et al., 2020). The research typically targets QG in the context of Question Answering, where the task is to generate a question that is related to the information in a given paragraph. The QA task ensures a general functional link between the question and the meaning of the passage that answers it. The datasets designed for such question answering/generation provide paragraph-level contexts for each question that span multiple sentences or even multiple passages. Note that the question here is related to the information expressed in the text passage, not to the way in which this information is structured and expressed in the text.

Consider the example from the SQuAD dataset shown in Figure 1. The first question pertains to the first sentence of the passage. While the concept *gravity* mentioned in that sentence is needed to answer the question, the question cannot be answered using the first sentence as such. For the second question, the information needed to answer the question is expressed in a sentence that is more in line with the question, but still falls short of the so-called question-answer congruence (Stechow, 1990; Sugawara, 2016) required for the sentence to serve as a direct answer to the question.

Context:	In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail.
Q_1:	What causes precipitation to fall? gravity
Q_2:	What is another main form of precipitation besides drizzle, rain, snow, sleet and hail? graupel

Figure 1: Example question-answer pairs from the SQuAD dataset (Rajpurkar et al., 2016)

Complementing QG in the prominent QA context, there are other strands of QG research that aim

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 24–34,
Aberdeen, Scotland, UK, 20-24 September 2021.

at generating questions that can be answered by a sentence as given in the text, putting a premium on question-answer congruence. This includes QG work in the educational application domain, where the perspective of the question is supposed to reflect the perspective of the author of a given text passage that the student is supposed to learn about (Heilman and Smith, 2010; Heilman, 2011; Rus et al., 2012). Recent work under this perspective includes Stasaski et al. (2021), who propose a neural question generation architecture for the generation of cause-and-effect questions. They extract cause and effect relations from text, which are then used as answers for the neural question generation, aiming at direct question-answer congruence.

A second strand of work for which the relation between the question and the answer sentence as expressed in the text plays a crucial role is the research interested in discourse. An early example of research investigating the role of discourse structure for question generation is Agarwal et al. (2011). They identify discourse relations in a text as cues motivating the generation of a question and then formulate questions that can be answered by the sentences with those discourse relations, while ensuring direct question answer congruence. In a related vein, approaches making use of so-called Questions under Discussion (QuDs) to identify the information structure of a sentence in a given discourse also rely on such a direct relationship between question and answer. In a recent paper pursuing this perspective, De Kuthy et al. (2020) show that a seq2seq based neural approach can successfully generate meaningful, well-formed questions that can function as Questions under Discussions in a formal theory of discourse. Similarly, Pyatkin et al. (2020) showed that using question-answer pairs obtained through crowdsourcing can be used reliably to annotate discourse. Based on their crowdsourced data, they train a pipeline of neural models to directly generate such question-answer pairs from text. The overall goal of question generation supporting discourse analysis is to generate a question for every sentence in a text to explicitly characterize the evolving discourse.

Viewed from the perspective of question generation for tasks requiring question-answer congruence, the QG task in essence consists of two steps: (i) replace the answer phrase in the source sentence with a matching question word and (ii) transform the rest of the sentence into a well-formed question. All the words that the generated question consists of are already given, so only the question word that matches the answer phrase needs to be generated anew. The sentence-question pair in example (1) taken from De Kuthy et al. (2020) illustrates this.

(1) A: Auch Otto Graf Lambsdorf ist **gegen zweierlei Wahlrecht.**
also Otto Graf Lambsdorf is against double voting rights
Otto Graf Lambsdorf is also against double voting rights.

Q: **Wogegen** ist auch Otto Graf Lambsdorf?
what against is also Otto Graf Lambsdorf
What is Otto Graf Lambsdorf against, too?

Except for the answer phrase *gegen zweierlei Wahlrecht* ('against double voting rights'), all words from the source sentence reappear in the generated question, including the named entity *Otto Graf Lambsdorf*. The only new material in the question is the question word *wogegen* ('what against').

While the text thus includes all the language needed to successfully generate the question, for seq2seq-based approaches based on word embeddings, the challenge arises that words present in the source sentence which do not appear in the material the embeddings were trained on are not adequately represented. As admitted in De Kuthy et al. (2020), unknown and rare words are therefore a problem and cannot be correctly generated in the question. Rare words are often replaced by semantically related words that are inappropriate in the given context.

In this paper, we explore an alternative: characters and subwords as form-based and more fine-grained representations of both the input and output of the question generation task. We will show that this avoids the unknown/rare word problem and results in a substantial improvement both in a quantitative BLEU evaluation and in terms of a qualitative analysis of the questions. Going beyond the particular QG task, our results contribute to the general endeavour of exploring the best choices of form or meaning-based input and output representations for neural approaches for a range of NLP tasks depending on their characteristics.

2 Related Work

Traditional question generation approaches that leveraged syntactic structures and linguistic features (Liu et al., 2010; Curto et al., 2012; Heilman, 2011) to define transformation rules on parse trees

are inherently limited in their scope and ability to deal with authentic language data. Deep learning has, in recent years, supplanted such methods given its ability to learn the syntactic and semantic properties and characteristics of language when provided with large amounts of natural language text.

Neural question generation is generally realised as a sequence learning problem, so a sequence-to-sequence (*seq2seq*) architecture (Sutskever et al., 2014) is a logical fit for this type of task. Here, the encoder network learns the latent representation of the source sentence and the decoder network generates the target question one word at a time. The work done by Du et al. (2017) introduces two such models, which are provided with the source sentence and paragraph-level information that encodes the context of the generated question. Borrowing from reinforcement learning, the work by Kumar et al. (2018) introduces policy gradients along with POS tags and named entity mentions to assign task-specific rewards to the training objective. Pointer-generator networks (Gu et al., 2016; See et al., 2017) with gated self-attention have been deployed to address the problem of rare and out-of-vocabulary words and larger contexts (Zhao et al., 2018).

The neural question generation models mentioned above, and many more in this vein, primarily focus on generating questions in English and consider words to be the atomic unit of meaning. They consequently approach the representation learning and text generation tasks at the word level. This assumption does not necessarily hold for languages such as Chinese, where the individual characters contain rich internal information. Neural language models that are trained on character-level inputs have been shown to capture more salient information about morphology than their word-level counterparts (Huang et al., 2016; Marra et al., 2018). Character-aware question answering systems (Golub and He, 2016; Lukovnikov et al., 2017) have similarly been shown to be resilient to the unknown word problem. To capture and combine information about language form and meaning, Bojanowski et al. (2017) proposed treating words as bags of character n-grams to enrich word embeddings with subword information. Byte-pair encoding (Shibata et al., 1999) has seen a recent resurgence in the context of generative language models where it is employed to perform subword segmentation without the necessity of tokenization or morphological analysis. Subword-level embeddings learned with the help of this method have been competitive in many downstream NLP tasks (Sennrich et al., 2015; Heinzerling and Strube, 2018; Xu et al., 2019).

To test performance and trade-offs between character-, subword-, and word-level representations in the context of question generation, we use the German question generation task proposed by De Kuthy et al. (2020), aimed at generating a Question under Discussion for each sentence in a discourse. The required question-answer congruence with the meaning and form requirements this entails, together with the relative morpho-syntactic richness and partially flexible word order of the German language make it an interesting experimental setting for exploring the potential advantages of character and subword representations.

3 Data

In terms of datasets for neural question generation models, contemporary approaches are generally trained on datasets created in the question answering context. These datasets, such as SQuAD (Rajpurkar et al., 2016), Quac (Choi et al., 2018), and Coqa (Reddy et al., 2019), are not well-suited for training models for tasks requiring high question-answer congruence, and they focus on English. Multilingual datasets like XQUAD (Artetxe et al., 2019), MLQA (Lewis et al., 2019), XNLI (Conneau et al., 2018), and TyDi QA (Clark et al., 2020) are similarly unsuitable as they contain only little data, intended as benchmark for the evaluation of question answering systems.

Given these limitations of the established English datasets for the research goals we are pursuing, we instead obtained the German QA answer corpus created by De Kuthy et al. (2020) and base our explorations on that dataset. The corpus contains 5.24 million sentence-question-answer triples which were generated by a transformation-based question generation system (Kolditz, 2015) on articles from the German newspaper *Die Tageszeitung (taz,* `http://taz.de`*)*. The corpus exhibits over 30 different types of questions, the most common of which are wh-questions asking for subject and object phrases (such as who or what questions in English) as well as various types of questions asking for adverbial modifiers (such as, for example, when or where questions). Some typical question-answer pairs will be discussed later in section 5.

4 Our Character and Subword-based Neural QG Approach

As the starting point and baseline of our approach, we take the same basic architecture as De Kuthy et al. (2020), a word-embedding based sequence-to-sequence model (Sutskever et al., 2014) with multiplicative attention (Luong et al., 2015). This was done in order to ensure comparability of our results with theirs. Furthermore, any fundamental changes to the neural architecture – such as using a Transformer (Vaswani et al., 2017) or a pointer-generator (Zhao et al., 2018) network – would make it more difficult to distinguish between any improvements offered exclusively by the change in input representation and those by the change in architecture.

To introduce character– and subword–level tokens, we defined an input pipeline consisting of the following steps: 1) UTF-8 text normalization was performed on the input sentence, 2) the normalized input sentence was parsed using *spaCy*'s (Honnibal et al., 2020) `de_core_news_sm` model to perform word-level tokenization and part-of-speech (POS) tagging, 3) a second tokenization pass was performed on each word token to generate character and subword tokens, and 4) each character and subword token pertaining to a given word token was assigned the latter's POS tag and the answer phrase indicator.

For character-level tokenization, each word was decomposed into a list of its component Unicode codepoints. Subword tokenization was performed with the *HuggingFace* Tokenizer library (Wolf et al., 2020). The library provides byte-pair encoding (BPE, Shibata et al., 1999) and unigram (Kudo, 2018) tokenization algorithms. BPE first constructs a baseline vocabulary with all unique symbols in a corpus. Then, merge rules that combine two symbols in the base vocabulary into a new symbol are learned iteratively until a desired final vocabulary size is reached. Conversely, unigram tokenization starts with a large initial vocabulary from which it repeatedly removes symbols that have the least effect on a loss function defined over the training data of a unigram language model. To reduce the size of the base vocabulary in both models, base symbols are directly derived from bytes rather than (all) Unicode codepoints. The library also includes the SentencePiece (Kudo and Richardson, 2018) algorithm, which processes the input as raw string sequences obviating the need for pre-tokenization.

Finally, bidirectional LSTM was used as the recurrent unit in the encoder as we expect the contextual information provided by the backward pass to not only enrich the sentential representation learned in the encoder but also lower the effective reduction in learnable parameters caused by the smaller vocabulary sizes of the character- and subword-level models. The per-timestep input to the encoder is the concatenation of the token embedding, POS embedding, and the answer phrase indicator. The final outputs of the encoder (hidden state, sequences, cell state) is the concatenation of the respective backward and forward layers of each output.

For the character-level models, a fixed-size vocabulary consisting of all the unique codepoints in the QA corpus was generated. Similarly, the subword tokenizers were trained on the entire corpus to generate vocabularies with 10K symbols each.[1]

5 Evaluation

For a comprehensive comparison, we trained five models: a word-level model to replicate De Kuthy et al. (2020), three subword models with different tokenization algorithms (byte-level BPE, SentencePiece BPE, and SentencePiece Unigram), and a character model. All models were trained on the same 400K training samples from the QA corpus for 20 epochs, and validation was performed on 40K samples. For each type of input representation, the model with the lowest validation loss was was evaluated on a held-out test set of 15K samples.

For their original model, De Kuthy et al. (2020) implemented a post-processing copy module to replace OOV marker tokens in the generated question with the original tokens from the source sentence; this behaviour was replicated for our word-level model. As model hyperparameters, we used: batch size: 128, encoder: Bi-LSTM, decoder: LSTM, encoder/decoder hidden size: 256/512, encoder/decoder dropout: 0.5, word/subword/character embedding dim: 300, decoder beam search width: 5. Table 1 shows the BLEU scores from comparing the ground-truth questions of the test set with corresponding model-generated questions. We used the standard *SacreBLEU* library (Post, 2018)[2] for the calculation of the BLEU scores.

[1] The subword vocabularies also include the base symbols found in the character vocabulary. In both cases, special meta tokens such as unknown, sentence-start and end markers were additionally added to each vocabulary.

[2] Version 1.4.10 with default parameters.

Model	BLEU 1/2/3/4	Cumulative
Word (De Kuthy et al., 2020)	93.8/86.5/81.0/76.5	84.24
Word (replication)	93.8/86.5/81.0/76.5	84.20
Subword (Byte BPE)	98.2/93.4/90.0/87.4	**91.97**
Subword (SentPiece BPE)	97.0/91.4/87.3/84.1	89.35
Subword (SentPiece Unigram)	98.1/93.3/89.8/87.2	91.76
Character	97.2/91.8/88.0/85.1	90.18
Subword-level (Byte BPE NoPOS)	98.0/93.0/89.4/86.7	91.48
Subword (SentPiece BPE NoPOS)	97.8/92.3/88.5/85.7	90.67
Subword (SentPiece Unigram NoPOS)	98.0/92.7/88.9/86.1	90.84
Character (NoPOS)	97.4/91.8/87.9/84.9	90.34

Table 1: Quantitative evaluation results

The word-level QG model with our modifications is able to produce results essentially identical to those of the baseline model by De Kuthy et al. (2020). Both models use the post-processing copy step to address the problem of out-of-vocabulary tokens, but neither is able to fully overcome it due to the intrinsic weaknesses of such extra-modular, non-neural solutions. The character- and subword-level models, on the other hand, entirely sidestep this issue by generating the target sequence one character or subword at a time. We additionally trained variants of the character- and subword-level models without POS tags (the NoPOS models in the table). Even with fewer learnable parameters and without the linguistic information provided by the POS tags, the models are able to achieve scores very close to those of their POS-aware counterparts. The effect of different subword tokenization algorithms on the quantitative performance of the model appears to be minimal.

5.1 Error Analysis

To analyze the quality of the results produced by our models and compare them to those of the baseline word-level model, we performed a manual evaluation of the questions generated for the same sample of 500 sentences of De Kuthy et al. (2020).

The quality of the generated questions was manually evaluated by two human annotators, both trained linguists and native speakers of German. They were asked to provide a binary judgment: whether the question is well-formed and satisfies question-answer congruence with the source sentence. The two conjoined criteria were expressed in the annotation manual as (i) Well-Formedness: Is the question grammatically correct and would I formulate it that way as a native speaker of German? and (ii) Question-Answer Congruence: Is the question answered by the associated sentence as a whole? The annotators were instructed to take into account all aspects of grammaticality, including word order, verb forms, punctuation, and also spelling and capitalization errors. For the evaluation of question-answer congruence, the annotators checked whether the generated question was answerable by the full source sentence, in particular whether the question word matched the given answer phrase and whether the question did not contain any semantically different words. The resulting annotation showed good inter-annotator agreement ($\kappa = 0.74$).

The results of this evaluation (Table 2) reveal how model performance increases with more fine-grained in input granularity. The baseline word-level model posted the worst score among all trained models, generating well-formed questions for only 54.2% of the 500 sentences in the evaluation set. The best subword model improves upon this substantially with 61.0% well-formed questions, and the character model adds a further, small improvement. Curiously, removing POS tags as input features from the subword model results in a slight performance increase, but the opposite for the character model. The effect is even more pronounced in the SentencePiece BPE subword model. To investigate this further, we performed systematic error analysis of the most frequently encountered errors (Table 3). Note that the overall sums differ slightly from the percentages in Table 2 since one question can contain multiple types of errors.

Model	Well-formed Questions
Word	54.2%
Subword (SentPiece Unigram)	59.6%
Subword (SentPiece Unigram No POS)	61.0%
Character	**61.4%**
Character (No POS)	59.6%

Table 2: Results per question for the evaluation set

Despite the post-processing copy mechanism, the questions from the word model still contained

Error Type	Word	Subword (SentPiece Unigram)	Subword (SentPiece Unigram NoPOS)	Character	Character (NoPOS)
Question word	82	108	100	109	117
Unknown Word	35	0	0	0	0
Word Order	29	20	23	21	23
Different Word	35	16	5	1	0
Different Subword	0	1	2	0	0
Missing Word	2	8	10	7	4
Missing Subword	0	0	2	0	0
Repeated Word	4	4	4	10	5
Verb Form	8	9	15	13	17
Source Sentence	13	13	13	13	13
Answer Phrase	23	31	31	24	26
Spelling	0	3	2	0	4
Total	231	217	205	197	213

Table 3: Distribution of error types in the evaluation samples

unknown words in 35 cases. For example, rare words such as *süffisant* (*smug*), *listenreich* (*cunning*), *Naschwerk* (*sweet delicacy*), *Erbtanten* (*rich aunt from which one inherits*). The subword and character models did not have this problem at all. Unwanted word replacements with different words occurred in 35 samples with the word model, for example, *unbegreiflich* (*incomprehensible*) was replaced by *geschehen* (*happen*), *Adelheid Streidel* (proper name of a terrorist) by *extremistischen Streidel* (*extremist Streidel*), and *bewilligt* (*approved*) by *beantragt* (*requested*). The subword models reduce this to as few as five occurrences, and in the character models this type of error does not occur at all. By far, the biggest error source for all models is the production of incorrect question words. This is a hard objective since the question word depends on aspects of form (e.g., does it refer to a nominal phrase or a prepositional phrase) and meaning (e.g., does it refer to an animate or inanimate referent) of the given answer phrase. The word-level model had fewer problems with question word generation than the other models, so the word embeddings encode sufficient form and meaning information for the model to learn the question word patterns.

There appears to be no single, clear pattern across all models that explains the effect of POS tags. Nevertheless, the quality of question words does consistently suffer when they are removed from the input. The character-based model without POS tags generated the highest number of questions with an incorrect question word - an aspect of question generation that relies on meaning-related information, evidently provided by the latter. One potential explanation could be rooted in how the models process the POS features: By assigning to each subword or character the POS tag of its parent word, the model has to contend with increased noise in the training data due to weak correlation between the tags and specific subwords or characters. For instance, the subword unit *her* in *herkommen (to come from)* would take the latter's POS tag `VB (verb)` but will be assigned `JJ (adjective)` when appearing in *herrlich (superb)*.

To gain a better understanding of when a model generates a new form and when it copies tokens from the input, in the following we discuss indicative examples together with the softmax-activated attention scores between the source sentence and the question. In the figures below, the x-axis and y-axis of each plot correspond to the tokens in the generated question and the source sentence, respectively. Each pixel corresponds to the alignment weight w_{xy} of the y-th source token and x-th target token, ranging from 0 (purple) to 1 (yellow). The red tokens on the y-axis indicate the phrase in the source sentence that answers the question.

Example (2) shows a typical sentence-question pair from the evaluation sample. Both the subword models and the character models produced the correct question in (2-b), given the answer phrase (marked in bold font).

(2) a. Bis dahin seien **die Länder der DDR** pleite.
until then would be the states of the GDR bankrupt

b. Wer ist bis dahin pleite?
who is until then bankrupt

For the correct question (2-b), the models have to produce the question word *Wer (who)* in place of the answer phrase *die Länder der DDR (the states of the GDR)*, and they have to transform

the plural *seien (were)* into the singular verb *ist (is)*. The sentence initial *Bis dahin (until then)* must be placed after the verb and lower cased. The attention plots in Figures 2 and 3 directly showcase this. The tokens in the answer phrase, particularly the first one, have higher alignment weights for the question word than the other tokens in the sentence. Similarly, the model specifically attends to the verb in the source sentence when generating the same in the question. The tokens that are copied as-is from the source sentence have strong, monotonic weights that appear as diagonals.

Example (3) shows another sentence-question pair from the evaluation set. The character model predicted the correct question (3-b), but the subword model predicted the incorrect question (3-c), in which the adverb *danach (thereafter)* is repeated and the numeral *1988* from the input is missing.

(3) a. Danach sollte Ende 1988 **mit der Produktion**
thereafter should end 1988 with the production
der U-Boote und mit der Teillieferung
of the submarines and with the partial deliveries
begonnen werden.
started be

Subsequently, production of the submarines and partial deliveries were to begin at the end of 1988.

b. Womit sollte danach Ende 1988 begonnen
with what should thereafter end 1988 started
werden ?
be

What should be started thereafter at the end of 1988?

c. Womit sollte danach Ende danach
with what should thereafter end thereafter
begonnen werden ?
started be

The corresponding attention scores are shown in Figure 4 for the correct question (3-b) generated by the character model and Figure 5 for the erroneous question (3-c) produced by the subword model. Once again, in order to produce the question word *Womit (with what)*, both models assign a strong weight to the preposition *mit (with)*, which is the first token of the given answer phrase. While the character model then continues to correctly annotate the tokens in the source sentence, the subword model's alignments show more ambiguity. For the token *danach (thereafter)*, it additionally attends to *Ende (end)* in the source sentence - another word that carries a temporal meaning. At the position of the numeral *1988*, the model assigns significant weights to all three temporally related words, but the weight for *Ende* is diminished due to its occurrence in the previous timestep. Neverthe-

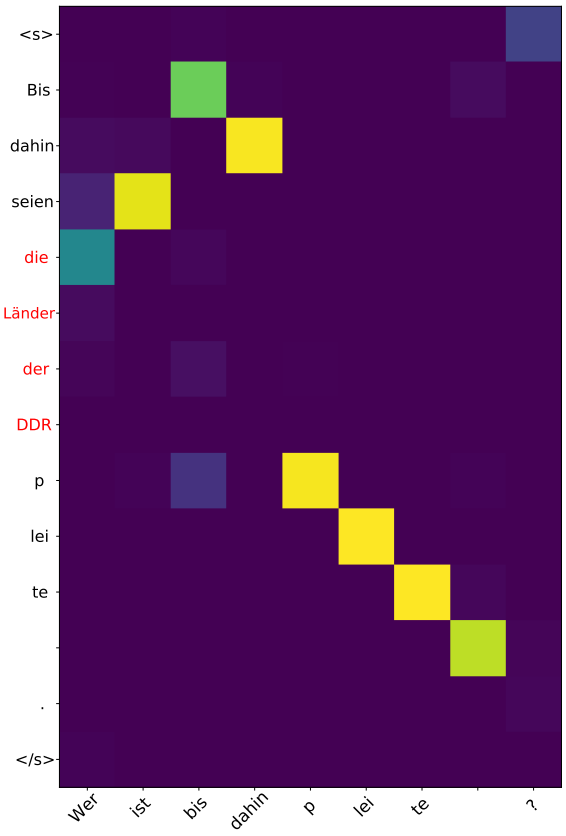


Figure 2: Subword attention plot for example (2)

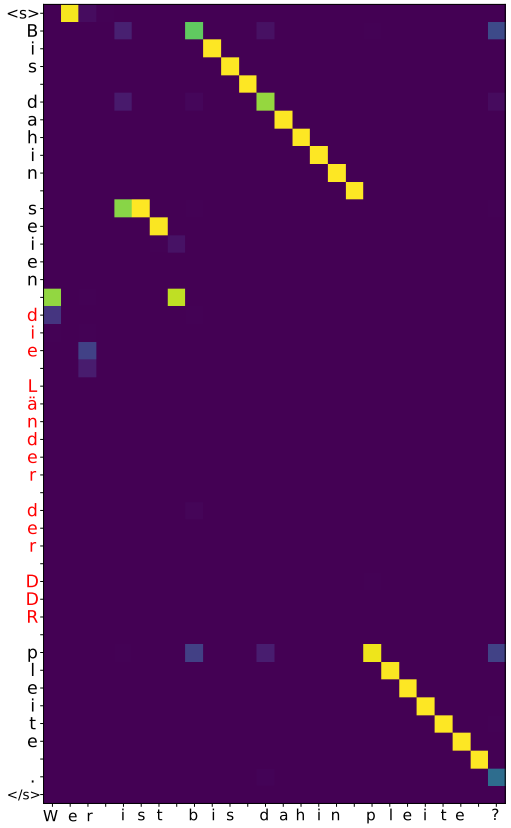


Figure 3: Character attention plot for example (2)

less, the model ultimately shows higher confidence in *danach* than in the rest and thus (mis-)predicts it a second time.

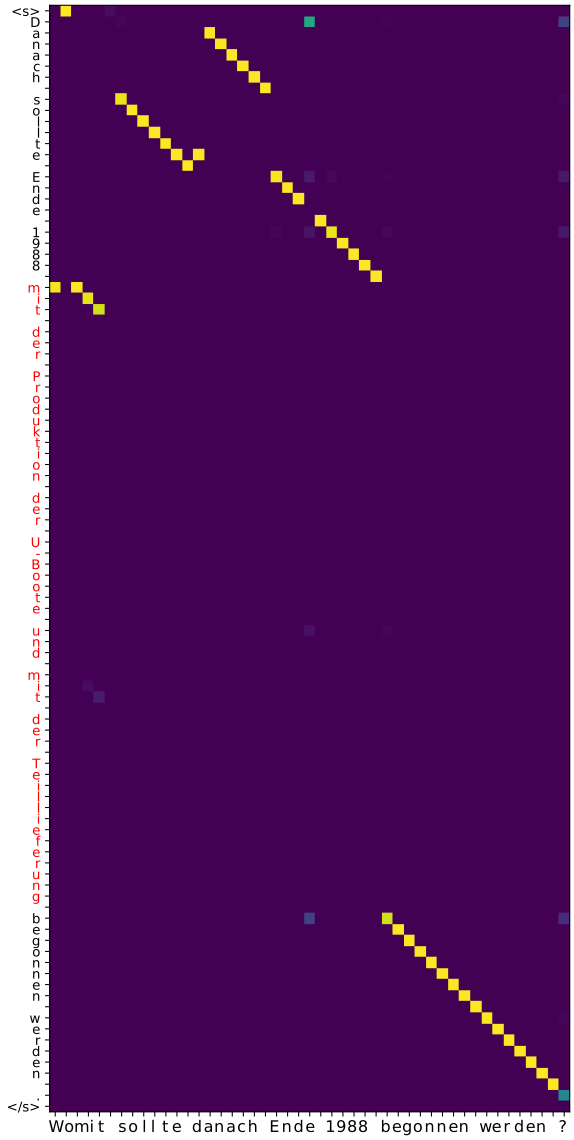


Figure 4: Character attention plot for example (3)

One potential problem of a purely form-based approach using characters is that it can produce character strings that do not correspond to any word in the given language. This hardly ever occurred in the questions generated by our character model with the exception of one interesting example where the model created a new question word, illustrated here in example (4).

(4) a. Dies dürfte sich mit der Schaffung des
this is likely itself with the creation of the
Binnenmarktes **ab** **1993** ändern.
single market from 1993 change
This is likely to change with the creation of the single market from 1993.

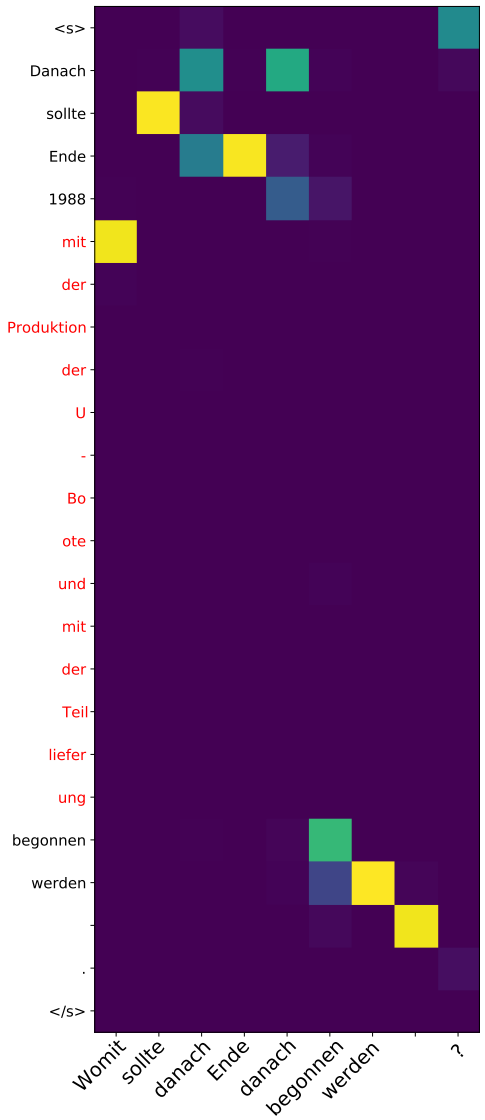


Figure 5: Subword attention plot for example (3)

b. Worab durfte sich dies mit der
where from was likely itself this with the
Schaffung des Binnenmarktes ändern?
creation of the single market change
From when was this likely to change with the creation of the single market?

Given the answer phrase *ab 1993*, the model produced the question word *Worab* – a concatenation of the (existing) words *wo (where)* and the preposition *ab (on)* – instead of the required question phrase *ab wann (from when)*. Such concatenations of a question word and a preposition actually exist in German, e.g., in the question word *woran (what of)*, so the character model apparently picked up this pattern of generating question words from prepositions, but applied it to a non-existing case.

6 Conclusion

We explored the prospect of neural question generation at the character- and subword-level using finer-grained input representations than word tokens by adopting De Kuthy et al. (2020)'s task of generating Questions under Discussion for German. The models that were trained on character and subword tokens showed significant leaps in BLEU scores in comparison to the baseline word-level model, even in the absence of extra linguistic information.

In addition to eliminating the problem of out-of-vocabulary and rare words, our manual analysis of the generated questions revealed that those models were able to learn and exploit both semantic and orthographic information with fewer parameters, producing questions with fewer errors relating to word order and word replacement. The character model, in particular, is able to fully eliminate the latter error category.

Considering the relevance of the research beyond the specific question generation task, the results reported in this paper provide further evidence and motivation to consider the advantages of form-focused neural representations and character-level natural language generation for tasks such as machine translation and extractive text summarization.

References

Manish Agarwal, Rakshit Shah, and Prashanth Mannem. 2011. Automatic question generation using discourse cues. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–9, Portland, OR. Association for Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2019. On the cross-lingual transferability of monolingual representations. *CoRR*, abs/1910.11856.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ying-Hong Chan and Yao-Chung Fan. 2019. A recurrent bert-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 154–162.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*.

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *arXiv preprint arXiv:2003.05002*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Sérgio Curto, Ana Cristina Mendes, and Luísa Coheur. 2012. Question generation based on lexico-syntactic patterns learned from the web. *Dialogue & Discourse*, 3(2):147–175.

Kordula De Kuthy, Madeeswaran Kannan, Haemanth Santhi Ponnusamy, and Detmar Meurers. 2020. Towards automatically generating questions under discussion to link information and discourse structure. In *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

David Golub and Xiaodong He. 2016. Character-level question answering with attention. *arXiv preprint arXiv:1604.00727*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Michael Heilman and Noah A. Smith. 2010. Extracting simplified statements for factual question generation. In *In Proceedings of the Third Workshop on Question Generation*.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Jiangping Huang, Donghong Ji, Shuxin Yao, Wenzhi Huang, and Bo Chen. 2016. Learning phrase representations based on word and character embeddings. In *Neural Information Processing*, pages 547–554, Cham. Springer International Publishing.

Tobias Kolditz. 2015. Generating questions for German text. Master thesis in computational linguistics, Department of Linguistics, University of Tübingen.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuan-Fang Li. 2018. A framework for automatic question generation from text using deep reinforcement learning. *arXiv preprint arXiv:1808.04961*.

Patrick Lewis, Barlas Oğuz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2019. Mlqa: Evaluating cross-lingual extractive question answering. *arXiv preprint arXiv:1910.07475*.

Ming Liu, Rafael A Calvo, and Vasile Rus. 2010. Automatic question generation for literature review writing support. In *International Conference on Intelligent Tutoring Systems*, pages 45–54. Springer.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pages 1211–1220.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Giuseppe Marra, Andrea Zugarini, Stefano Melacci, and Marco Maggini. 2018. An unsupervised character-aware neural approach to word and context representation learning. *Lecture Notes in Computer Science*, page 126–136.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Valentina Pyatkin, Ayal Klein, Reut Tsarfaty, and Ido Dagan. 2020. QADiscourse - Discourse Relations as QA Pairs: Representation, Crowdsourcing and Baselines. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2804–2819, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2012. A detailed account of the first question generation shared task evaluation challenge. *Dialogue & Discourse*, 3(2):177–204.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. 1999. Byte pair encoding: A text compression scheme that accelerates pattern matching. Technical report, Technical Report DOI-TR-161, Department of Informatics, Kyushu University.

Katherine Stasaski, Manav Rathod, Tony Tu, Yunfang Xiao, and Marti A. Hearst. 2021. Automatically generating cause-and-effect questions from passages. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 158–170, Online. Association for Computational Linguistics.

Arnim von Stechow. 1990. Focusing and backgrounding operators. In Werner Abraham, editor, *Discourse Particles*, pages 37–84. John Benjamins, Amsterdam.

Ayaka Sugawara. 2016. *The role of question-answer congruence (QAC) in child language and adult sentence processing*. Ph.D. thesis, Massachusetts Institute of Technology.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on*

Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Yuxi Xie, Liangming Pan, Dongzhe Wang, Min-Yen Kan, and Yansong Feng. 2020. Exploring question-specific rewards for generating deep questions.

BinChen Xu, Lu Ma, Liang Zhang, HaoHai Li, Qi Kang, and MengChu Zhou. 2019. An adaptive wordpiece language model for learning chinese word embeddings. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 812–817. IEEE.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910.

Towards Zero-Shot Multilingual Synthetic Question and Answer Generation for Cross-Lingual Reading Comprehension

Siamak Shakeri [†]
Google Research
siamaks@google.com

Noah Constant
Google Research
nconstant@google.com

Mihir Sanjay Kale
Google Research
mihirkale@google.com

Linting Xue
Google Research
lintingx@google.com

Abstract

We propose a simple method to generate multilingual question and answer pairs on a large scale through the use of a single generative model. These synthetic samples can be used to improve the zero-shot performance of multilingual QA models on target languages. Our proposed multi-task training of the generative model only requires labeled training samples in English, thus removing the need for such samples in the target languages, making it applicable to far more languages than those with labeled data. Human evaluations indicate the majority of such samples are grammatically correct and sensible. Experimental results show our proposed approach can achieve large gains on the XQuAD dataset, reducing the gap between zero-shot and supervised performance of smaller QA models across various languages.

1 Introduction

Generating question and answers from raw text has always been a challenging problem in natural language generation. Recently, there have been numerous efforts around question generation (Du et al., 2017; Song et al., 2018; Klein and Nabi, 2019; Wang et al., 2020; Ma et al., 2020; Chen et al., 2020; Tuan et al., 2019).

Using such synthetic samples to improve the performance of question answering models has been explored by Puri et al. (2020), Alberti et al. (2019), and Shakeri et al. (2020), who show that reading comprehension (RC) models can be improved by generating large-scale synthetic training data. These promising results combined with the recent surge in the development of powerful generative models such as GPT-3 (Brown et al., 2020), BART (Lewis et al., 2020a), and T5 (Raffel et al., 2020) suggest that the need for large manually labeled datasets can be reduced.

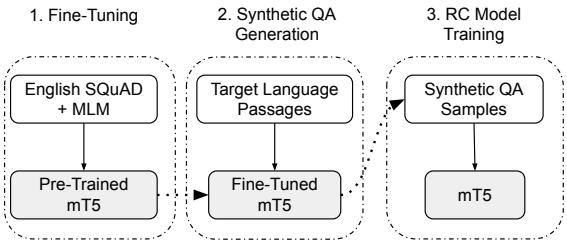


Figure 1: End-to-End pipeline: 1) Fine-tuning the generative model using SQuAD English samples and multilingual MLM. 2) Generating synthetic samples from Wikipedia passages of the target language using the fine-tuned generative model. 3) Training the downstream reading comprehension model using synthetic samples.

Although synthetic question-answer (QA) generation is well explored in English, the efficacy of such methods in the other languages remains an open question. Considering the lack of manually labeled QA datasets in many languages other than English, QA generation techniques can be applied to improve RC models in those languages. The emergence of multilingual generative models such as mBART (Liu et al., 2020a) and mT5 (Xue et al., 2021) facilitates such endeavors.

In this work, we propose generating multilingual question answer pairs to improve the performance of RC models in languages other than English. Besides unlabeled articles and questions, our proposed method only requires labeled training samples in English, thus completely removing the need to acquire new labeled datasets. Our approach can easily be extended to any language, as long as the multilingual generative model supports the language, and unlabeled questions and articles, such as Wikipedia, books, etc., exist in that language.

To enable zero-shot QA generation, the generative model should be able to produce non-English QA samples from non-English inputs when only

[†] Corresponding author.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 35–45,
Aberdeen, Scotland, UK, 20-24 September 2021.

trained on English samples. Inspired by the work of Artetxe et al. (2020); Gururangan et al. (2020); Liu et al. (2020b), we propose a multi-task learning setting, where during the fine-tuning stage, we train on two tasks in parallel: the target question-answer generation task, and the multilingual masked language modeling (MLM) task that was used in pre-training the generative model. Our experimental results show that including the MLM task is crucial in enabling the zero-shot capability of the fine-tuned generative model.

We propose fine-tuning a pre-trained multilingual T5 model on the SQuAD 1.1 (Rajpurkar et al., 2016) training set. The fine-tuned model is then used to generate a large set of synthetic question-answer pairs from Wikipedia passages in the target language. Fig. 1 illustrates the end-to-end pipeline. We show that such synthetic samples can significantly boost RC models trained only on the English samples, with improvements up to 9 absolute points on F1. To summarize, our contributions are:

- Improving the zero-shot performance of multilingual RC models on multilingual QA tasks through generation of synthetic multilingual QA pairs.
- Proposing a multi-task fine-tuning of the multilingual generative model which is crucial for enabling zero-shot multilingual generation.
- Our approach is entirely zero-shot. No manually-labeled sample is used in fine-tuning the generative model on target languages, making our method applicable to both high and low resource languages.
- Demonstrating grammatical correctness and sensibility of generated questions through human evaluations.

The rest of the paper is organized as follows. In section 2, we discuss the process designed to train the generative model and produce synthetic samples. Section 3 discusses related work in the area of multilingual question-answer generation. In section 4, we present experiments to measure the quality of generated samples. Section 5 focuses on the application of synthetic question-answer samples to downstream reading comprehension models. Finally, we conclude in section 6.

2 End-to-End Question-Answer Generation and Filtering

2.1 Modeling

We use pre-trained "multilingual T5" (mT5) (Xue et al., 2021) as our generative model. The mT5 model is based on T5 (Raffel et al., 2020), which is an encoder-decoder sequence-to-sequence model.

2.2 QA Generation Task

We follow the probability distribution factorization suggested by Shakeri et al. (2020), where:

$$p(Q, A|P) = p(Q|P) \times p(A|Q, P)$$

Sampling from the above factorization is performed as follows:

$$q \sim p(Q|P), a \sim p(A|Q, P)$$

where Q, P, A refer to question, passage, and answer, respectively. During fine-tuning, passage tokens are fed as inputs, and the targets are a concatenation of the question and answer tokens. During sampling, candidate passages are passed as inputs to the fine-tuned generative model, and question-answer pairs are sampled from the decoder.

Fig. 2 depicts the fine-tuning and sampling processes. We prepend *"question"* to the question tokens and *"answer"* to the answer tokens, to help the model distinguish one from the other.

2.3 Masked Language Modeling Task

The mT5 model is pre-trained on the large multilingual "mC4" dataset (Xue et al., 2021) built from Common Crawl data, and trained using a Masked Language Modeling (MLM) task. This task involves replacing contiguous spans of input tokens with unique sentinel tokens (one per span). The decoder is then trained to reconstruct all the masked spans in the input, using a standard cross-entropy loss with teacher forcing. We use a variant of this MLM task, where we remove all "sentinel" tokens (corresponding to non-masked spans in the input text) from the target sequence, as we find this improves the quality of generated QAs.

2.4 Multi-Task Fine-Tuning

To perform zero-shot generation, the model needs not only to learn the QA Generation task, but also to retain its multilingual generation capabilities achieved during pre-training. To avoid *catastrophic*

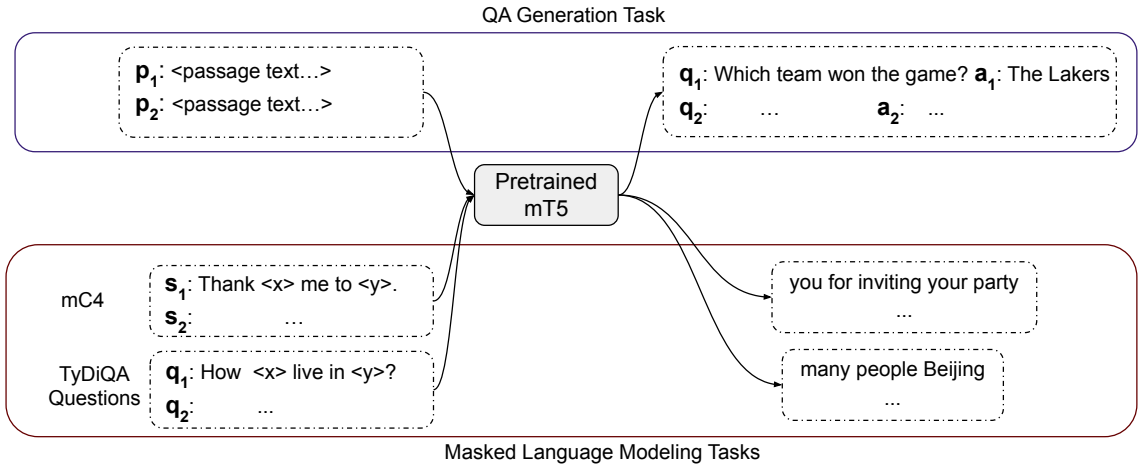


Figure 2: Multi-task fine-tuning of the multilingual pre-trained mT5 model. 1) QA generation task, which uses SQuAD English samples. 2) MLM task on a subset of mC4. 3) MLM on only the questions from the TyDiQA Gold Passage Task. The MLM variant used does not include sentinel tokens in the decoder output.

forgetting (French, 1999), which could lead to degraded generation capability, we propose a multi-task setting, where a predetermined percentage of fine-tuning examples come from the QA Generation task, while the remaining examples (trained in parallel) are from a mixture of two MLM tasks: 1) MLM on a subset of mC4 which is a continuation of mT5 pre-training, 2) MLM on only the questions from TyDiQA Gold Passage dev and training sets. The MLM task on mC4 helps the fine-tuned model retain its multilingual generation capabilities, while the MLM task on TyDiQA questions further improves the question generation capabilities of the generative model. Note that the only supervised QA training data is SQuAD 1.1. The MLM task on TyDiQA questions is not conditioned on the associated passages of the questions. Experimental results in section 4 demonstrate the efficacy of our proposed approach. Fig. 2 illustrates the multi-task fine-tuning process.

Fig. 3 demonstrates examples of generated samples in five languages using an mT5-XL (3.7B parameter) model fine-tuned in the multi-task setting (§2.4). It can be observed that: 1) the generated questions are in the same language as the passage most of the time, 2) the answers are relevant to the generated questions, 3) the model is capable of generating long and non-trivial QA pairs.

Fig. 4 illustrates generated QA samples in Spanish and Arabic, when only the QA Generation task (§2.2) is included in the fine-tuning. We observe: 1) questions are primarily in English, not the target language, 2) outputs contain certain tokens and entities mentioned in the language of the passage, and 3) ignoring language issues, the outputs exhibit semantically well-formed QA correspondence.

2.5 Decoding and Filtering

Since the quality of the generated question answer pairs is vital in improving the performance of downstream models, the generated samples require a strong filtering technique. Using the F1 score of a trained RC model to perform filtering, a.k.a. *round-trip* filtering, has been previously explored by Puri et al. (2020) and Alberti et al. (2019). For a generated QA sample *(q, a, p)*, where q, a, and p indicate question, answer, and passage, the following steps are performed: 1) a trained RC model is applied to *(q, p)*, predicting a', and 2) the F1 score of a and a' is calculated, and if above a certain threshold, *(q, a, p)* is kept, otherwise dropped.

3 Related Work

Recent work has explored question-answer generation (Alberti et al., 2019; Puri et al., 2020; Lee et al., 2020; Shakeri et al., 2020), but limited in scope to English. We leverage the modeling and filtering approaches proposed by Shakeri et al. (2020) due to their simplicity and effectiveness.

Kumar et al. (2019) explores cross lingual question generation. In contrast to our work, this only generates questions, without the corresponding answers. Additionally, this approach requires a complicated pre-training process on the target languages, as well as gold samples to fine-tune the generative models, so it is not easily extensible to

Spanish

Temprano en la mañana, cuando hay poco tráfico en las autopistas Tōmei y Chūō, el viaje entre Tokio y el lago Yamanaka toma una hora. El tráfico se vuelve pesado más tarde y el viaje toma un tiempo más largo. La ruta nacional de Japón 138 y la ruta nacional de Japón 413 corren a lo largo de las orillas del lago. Hay servicios de autobús disponibles desde la estación Fujisan (línea Fujikyuko), la estación JR Gotemba (línea JR Gotemba) y la estación JR Mishima (JR Shinkansen). Los servicios de autobuses de la autopista también están disponibles en la estación Shinjuku (Tokio).

Q: *¿Cuánto tiempo se lleva entre Tokio y el lago Yamanaka?* A: *una hora*
Trans.: *How long does it take between Tokyo and Lake Yamanaka?* *one hour*

Arabic

في جيز الدي يصنعها الهاوس اغاني أول وكانت العشرين. القرن ثمانينات بداية في الأمريكية شيكاغو مدينة في وتداولها انتاجها بدأ الالكترونية الرقص موسيقى انواع من نوع هي الهاوس موسيقى والذي الهاوس أغاني انتاج احترف من أول بالمرقص المقيم جي الدي كان الذي ناكليس فرانكي الموسيقى.يعتبر هذه اسم اشتق ومنه هاوس" "ووتر مرقص في تبث الموسيقي النمط هذا من المدينة الفانك. موسيقى في المستخدمة الأصوات من قريب وبصوت منخفض، بإيقاع يمتاز جديد موسيقي نمط وضع اختار

Q: ما هو نوع من انواع موسيقى الرقص الالكترونية التي بدأ انتاجها وتداولها في مدينة شيكاغو الأمريكية في بداية ثمانينات القرن العشرين؟ A: موسيقى الهاوس
Trans.: *What kind of electronic dance music was produced and circulated in the American city of Chicago at the beginning of the eighties of the twentieth century?* *house music*

Chinese

玩具总动员《玩具总动员》() 是一部于1995年上映的美国计算机动画冒险喜剧片，由皮克斯动画工作室制作，华特迪士尼影业发行。该片是约翰·拉塞特的导演处女作，也是皮克斯制作的一部计算机动画长片和第一部剧场影片。影片的主角是一队在人类面前假装没有生命的拟人化玩具，侧重于提线牛仔玩偶（汤姆·汉克斯配音）和宇航员巴斯光年（提姆·艾伦配音）之间的关系。影片由约翰·拉塞特、安德鲁·斯坦顿、、和乔斯·韦登编剧，兰迪·纽曼配乐。史蒂夫·乔布斯和艾德文·卡特姆担纲执行制片人。影片于1995年11月22日在剧院上映。为宣传他们的计算机而制作动画短片的皮克斯在以小玩具讲故事的短片《小锡兵》大获成功后，接触迪士尼，打算制造一部计算机动画长片。拉塞特、斯坦顿和皮特·多克特将先前写好的剧本大纲交给迪士尼，推动一部更前卫的电影。故事卷轴被毁灭后，制作被中断，剧本被重写，皮克斯所希望的“玩具深深希望孩子们能和它们一同玩耍，这种欲望驱动着它们的希望、恐惧和行动”的主题和基调得到更好地反映。工作室之后组建了一支员工相。

Q: 《玩具总动员》是什么类型的电影? A: 美国计算机动画冒险喜剧片
Trans.: *What kind of movie is “Toy Story”?* *American Computer Animation Adventure Comedy Movies*

Russian

В местном кметстве Флорентин, в состав которого входит Флорентин, должность кмета (старосты) исполняет Любка Георгиева Константинова (коалиция в составе 2 партий: Болгарская социалистическая партия (БСП), Земледельческий союз Александра Стамболийского (ЗСАС)) по результатам выборов правления кметства.Кмет (мэр) общины Ново-Село —Георги Герасимов Стоенелов (независимый) по результатам выборов в правление общины.

Q:*Кто управляет общиной Ново-Село?* A:*Георги Герасимов Стоенелов*
Trans.: *Who runs the Novo-S community?* *Georgi Gerasimov Stoene*

Hindi

सिन्धी भाषा विधेयक, १९७२ पाकिस्तान के सिन्ध विधानसभा में १९७२ में प्रस्तुत एक विधेयक (बिल) था जिसे ३ जुलाई को तत्कालीन मुख्यमन्त्री मुमताज भुट्टो ने प्रस्तुत किया। इस विधेयक ने सिन्ध प्रान्त में सिन्धी भाषा को एकमात्र आधिकारिक भाषा घोषित कर दिया। इसके बाद ७ जुलाई से सिन्ध में भाषाई दंगे शुरू हो गये। तब पाकिस्तान के प्रधानमन्त्री जुल्फिकार अली भुट्टो ने घोषणा की कि उर्दू और सिन्धी दोनों ही सिन्ध की राजभाषाएँ होंगी। आधिकारिक रूप से सिन्धी को उर्दू के समकक्ष बनाये जाने से उर्दू बोलने वाले लोग निराश हो गये क्योंकि वे सिन्धी नहीं बोलते थे।

Q: किस भाषा को एकमात्र आधिकारिक भाषा घोषित कर दिया ? A: सिन्धी
Trans.: *Which language is declared as the only official language?* *Sandhi*

Figure 3: Samples of generated QAs in Spanish, Russian, Chinese, Arabic, and German. The generative model is mT5-XL fine-tuned on the mixture setting of section 2.4. ***Trans.*** refers to translations of the QA sample using Google Translate service.

other languages. This is in contrast to our approach, which does not require any gold QA samples in any language other than English. Another distinguishing factor is that we demonstrate improved performance on downstream QA tasks, while Kumar et al. (2019) only measure the quality of the generated samples on automatic metrics such as BLEU, and human evaluations.

Similarly, Chi et al. (2020) explore cross-lingual question-only generation using SQuAD English samples. They propose cross-lingual pre-training on the source and target languages. Similar to Kumar et al. (2019), their focus is only on the quality of the generated questions, whereas we validate our approach directly through improvements on downstream QA tasks. Moreover, while Chi et al. (2020) depends on a complex pre-training recipe and parallel sentences in both source and target languages, our approach not only does not require such parallel corpus, but also the MLM task included in our fine-tuning setting is widely used and studied. This leads to our approach being more easily adaptable to other languages and pre-trained generative models.

Most closely related to our work is the multilingual synthetic question generation approach of Riabi et al. (2020). However, there are two important differences between the two approaches. Firstly, our work includes both question and answer generation using a single model, while theirs only focuses on question generation. We believe generating the question and answer jointly is a richer problem that better harnesses the capabilities of pre-trained language models. Their question generation is conditioned on the selected answers, which further limits the generation. Secondly, their proposed method depends on translating SQuAD to target languages to fine-tune the generative model, hence limiting the application of their approach to languages where such translation data exists. Furthermore, even when translated data is available, the quality of samples generated by a model trained on such data is highly affected by the quality of the translations. This could lead to low quality QA samples in low resource languages. This is in contrast to our zero-shot approach, which does not require any training data in the target language.

Spanish

Richard Anton Patrick Connel O'Ferrall (Paramaribo, 21 de julio 1855 - 28 de octubre de 1936) fue un maestro, escritor y miembro del Parlamento de Surinam. Durante la última década del siglo XIX y principios del siglo XX O'Ferrall fue una de las figuras centrales en la vida cultural de Paramaribo. Se graduó de maestro en holandés, francés e inglés y en 1881 escribió acerca de un método para la enseñanza de la lectura inicial. Fue director de una escuela especial para la Educación Primaria Avanzada (ULO) y también director de la puesta en marcha en 1888 de la escuela pública de formación de artesanos que publicó desde 1893 un catálogo de obras y técnicas constructivas. Con los estudiantes de la clase más alta que formó el grupo teatral Ons Genoegen. Él escribió artículos para la revista De Ambachtsman (El Artesano), pero en ocasiones también para los periódicos locales.

Q: *Who was Richard Anton Patrick Connel O'Ferrall?* A: *maestro, escritor y miembro del Parlamento de Surinam*
Q: *When did Richard O'Ferrall die?* A: *28 de octubre de 1936*

Arabic

آبلة (بالإسبانية: Ávila) هي مدينة تقع في وسط إسبانيا، هي عاصمة مقاطعة آبلة التابعة لمنطقة قشتالة وليون.الديموغرافيا بلغ عدد سكان مدينة آبلة 008.59 نسمة عام 2011 (وفقاً للمعهد الوطني للإحصاء الإسباني).توأمة لآبلة اتفاقيات توأمة مع كل من: فيلنوف سور لوت روي-ماليزون تيرامو رودسأعلام توماس دي توركيمادا خيسوس هرنانديز أوييدا سانتشث ألبورنوث فيليسيانو ريفيلا خيسوس هرنانديز جيل غونزاليس جيل غونزاليس دافيلا توماس لويس دي فيكتوريا كارلوس ساستري خوليو خيمينيزمراجعانظر أيضًا قائمة مواقع التراث العالمي في إسبانيا موقع التراث العالمي قائمة ...

Q: *Where is* آبلة *located?* A: فوسط إسبانيا
Q: *How many people lived in* آبلة *in 2011?* A: *59.008*

Figure 4: Samples of generated QAs in Spanish and Arabic. The mT5-XL model is unable to generate valid questions in the target language, as in this case it was fine-tuned exclusively on the English QA generation task from section 2.2.

4 Experimental Setup and Results

4.1 Datasets

SQuAD (Rajpurkar et al., 2016) is an English QA dataset consisting of 100k samples. The passages are extracted from Wikipedia. We use the train and dev splits of SQuAD 1.1 in this work.

XQuAD (Artetxe et al., 2020) is a multilingual QA dataset consisting of 240 paragraphs and 1190 question-answers in Arabic, Chinese, German, Greek, Hindi, Russian, Spanish, Thai, Turkish and Vietnamese. These samples have been professionally translated from the SQuAD 1.1 dev set.

MLQA (Lewis et al., 2020b) is a benchmark dataset for evaluating cross-lingual question answering performance. This dataset contains over 5k QA instances (12k in English) following the SQuAD format in each of Arabic, Chinese, English, German, Hindi, Spanish and Vietnamese. We use the test split in our evaluations.

TyDiQA (Clark et al., 2020) is another multilingual QA dataset consisting of 200k QA pairs from 11 typologically diverse languages. There is less lexical overlap between questions and answers compared to XQuAD and MLQA. We use the Gold Passage task, which includes ∼50k samples in the train split and between 130 and 1,100 samples for each language in the development set.

XTREME (Hu et al., 2020) is a multilingual benchmark consisting of nine tasks spanning 40 typologically diverse languages. This dataset includes machine translated SQuAD 1.1 train and dev samples, which we employed in our experiments. We refer to such samples as `translate-train`.

4.2 Generative Model Fine-Tuning

We used the official mT5-XL model (Xue et al., 2021) with 3.7 billion parameters as our generative model. The official pre-trained checkpoint is fine-tuned using the mixture of tasks described in section 2.1. We chose the task mixing ratio to be 10:1, meaning for every 10 instances of the QA Generation task (§2.2), we mix one instance of the MLM task (§2.3). We experimented with mixing ratios of 100:1 and 1000:1 as well, both of which under-performed 10:1. The unsupervised MLM task covers text from two domains: 1) the subset of the mC4 corpus (Xue et al., 2021) covering Arabic, Bengali, English, Finnish, Indonesian, Korean, Russian, Swahili, and Telugu, and 2) questions from TyDiQA (Clark et al., 2020) train and dev sets, covering the same set of languages.

It is worth highlighting that we only fine-tune a single model to generate across all target languages. We do not apply language code prompts during fine-tuning or inference. We observe that by properly designing the fine-tuning mixture, the model is capable of generating samples that match the language of the input passage. Human evaluations in section 4.4 further verify this.

All of our models are fine-tuned for 5,000 steps with a batch size of 131,072 tokens, distributed over 64 TPU-v3 chips. We use the Adafactor optimizer (Shazeer and Stern, 2018) with constant learning rate of 1e-3. The final checkpoint is used to perform synthetic data generation.

4.3 Automatic Evaluation Results

To compute automatic metrics such as BLEU against QA samples of the development set, we modify the generation task to generate a question

Training Task	ar	de	en	es	hi	vi	zh
SQuAD en	1.7	3.0	23.4	3.6	3.2	4.4	1.2
Mixture 1	12.2	14.9	**25.0**	18.4	10.6	13.8	**10.0**
Mixture 2	13.1	**15.2**	24.9	18.4	**11.1**	**13.9**	9.7
Mixture 3	**14.5**	14.8	**25.0**	**18.6**	10.8	13.5	9.6

Table 1: Comparison of question generation quality (BLEU score) on the MLQA test set with mT5-XL: The Mixtures are as follows: *SQuAD en*: SQuAD en as the training data, *Mixture 1*: SQuAD en + MLM on mC4 subset, *Mixture 2*: SQuAD en + TyDiQA questions, *Mixture 3*: SQuAD en + MLM on mC4 subset + MLM on TyDiQA questions.

Model Size	ar	de	en	es	hi	vi	zh
Base (580M)	3.9	5.1	19.0	8.2	3.5	7.4	3.1
Large (1.2B)	10.3	5.7	23.9	5.9	4.3	6.2	3.9
XL (3.7B)	14.5	14.8	**25.0**	18.6	10.8	13.5	9.6
XXL (13B)	**15.8**	**16.2**	**24.9**	**19.3**	**12.2**	**15.6**	**10.2**

Table 2: Performance of question generation (mixture setting) on the MLQA test set for different mT5 model sizes.

given the passage and answer. Conditioning on the answer is needed, as without it, the generative model might generate samples that are of high quality but not related to the answers provided in the development set for a given passage. This would lead to difficulty in interpreting metrics such as BLEU.

Tab. 1 compares BLEU[1] performance of two fine-tuning settings on the MLQA test set. We report results using the mT5-XL model. As can be seen, including the MLM tasks has a large impact on performance, conveying large gains up to +15 absolute BLEU points. This is in line with our observations from section 2.5, where adding MLM fine-tuning task enabled the generative model to produce QA samples in the language of the target passage.

Interestingly, MLM on either mC4 or TyDiQA questions results in similar BLEU scores. Furthermore, using a mixture of the two does not yield additional gains. However, eyeballing the generated samples indicated that the model fine-tuned on the mixture of both MLM tasks and the supervised English task generates more well-structured and sensible questions and answers. Human evaluations in section 4.4 verify the high quality of generated samples from a model trained with this mixture.

To investigate the effect of the generative model size on the quality of data generation, we perform experiments using mT5 variants with different number of parameters: Base (580M), Large (1.2B) and XL (3.7B). We report results of the fine-tuned models with the mixture setting (§2.4) on the MLQA dataset in Tab. 2. Model performance improves dramatically with the size of the pre-trained model. Based on these results, for the remainder of the paper, we use the mT5-XL model fine-tuned using the mixture approach.

4.4 Human Evaluations

To perform manual quality evaluation of the generated questions, raters were presented with generated questions, and tasked with rating them according to the following criteria:

- *Is the question in the target language?* Raters could select *yes* or *no*.
- *grammatical correctness:* Raters could select a whole number from 1 (lowest) to 4 (highest).
- *sensibility:* Raters could select a whole number from 1 (lowest) to 4 (highest).

In total, 400 generated samples from 5 languages were randomly selected and rated by native speakers of each language. Each rater was assigned 40 samples. Two native speakers of each of the five languages were asked to perform the task. Tab. 3 shows the evaluation results.

The results show that the multilingual generative model is nearly perfect at generating samples that match the language of the input passage. Considering no language codes are used during fine-tuning, and only English supervised training data are used, the results show that our proposed mixture has enabled the model to perform zero-shot cross-lingual generation coherently.

Interestingly, Spanish samples achieve high scores in all of the categories. Considering the model is not trained on any Spanish samples, either in the MLM tasks or SQuAD 1.1, the model shows strong transfer learning capabilities. This implies that including the MLM task as proposed in our mixture setting not only prevents the generative model from catastrophic forgetting of its multilingual capability on the languages included in the MLM fine-tuning task, but also on those not included. The same argument partially applies to Hindi. While there are no Hindi samples in the fine-tuning mixture, Bengali (a related Indo-Aryan language) was seen in the MLM task.

[1] All BLEU scores in this work are calculated using SacreBLEU v1.3.0 (Post, 2018), with "exp" smoothing and "intl" tokenization.

	Target Language	Grammatical Correctness	Sensibility
Arabic	0.98	3.55	3.35
Chinese	1.00	3.60	3.60
Hindi	1.00	2.93	3.35
Russian	1.00	3.50	3.75
Spanish	1.00	3.10	3.05
Average	1.00	3.34	3.38

Table 3: Human evaluation metrics on the generated samples. Samples are randomly drawn, and rated by native speakers. "Target Language" scores are in the range 0–1, while the other columns range from 1–4.

5 Application of Synthetic Data to Multilingual Reading Comprehension

In this section, we describe experimental results that demonstrate the efficacy of using synthetic samples for improving multilingual reading comprehension (RC) models. This refers to the setting where given a passage and a question, the model is tasked with finding a span of the passage that answers the question.

5.1 Synthetic Data Generation

We randomly selected 10k paragraphs from Wikipedia, for each of Arabic (ar), German (de), Hindi (hi), Russian (ru) and Spanish (es). The selected paragraphs were restricted to have between 30 and 450 tokens, thereby removing passages that are too long or too short.

We fine-tune the mT5-XL model according to the mixture setting discussed in section 2.4 and the hyper-parameters from section 4.2, and then use this model to generate 20 questions per passage. We apply *top-k* sampling (Holtzman et al., 2020) with *k=10* and temperature of 0.5. The generated samples are processed to ensure: 1) each consists of a question followed by an answer, 2) the answer does exist in the passage. This was done to ensure answers are extractive. Non-extractive or no-answer QA are outside the scope of this work.

Finally, as discussed in section 2.5, round-trip filtering is applied to the generated QA samples. We use an mT5 XL model trained on SQuAD 1.1 (Rajpurkar et al., 2016) as the filtering model. The overall process results in approximately 10-20k synthetically generated samples in each target language. These generated samples are then used for training the RC models.

5.2 RC Model Fine-tuning

All of our reading comprehension models are initialized from the official mT5 (Xue et al., 2021) and later fine-tuned on the generated samples. We experimented with Base (580M), Large (1.2B), and XL (3.7B) parameter variants of mT5. We fine-tune using the TensorFlow framework. Each model was trained for 10,000 steps with a learning rate of 1e-3 and a batch size of 131,072 tokens. The models were trained on 16 TPU-v3 chips. In experiments where both the SQuAD 1.1 samples and synthetically generated samples are used to fine-tune the RC models, the model is trained on a mixture of the two, with a 1:1 mixing ratio. Adafactor optimizer (Shazeer and Stern, 2018) with constant learning rate of 1e-3 is used in all cases.

5.3 Results

Tabs. 4–6 demonstrate the F1 performance of the RC models trained on SQuAD 1.1 samples as well as synthetic data generated as described in 5.1 on mT5 Base, Large, and XL models. "*SQuAD en* " refers to the original SQuAD 1.1 (Rajpurkar et al., 2016) dataset in English. Our zero-shot baselines (denoted "ours") were slightly higher than those reported in Xue et al. (2021) (denoted "paper").

We observe that augmenting *SQuAD en* with synthetic samples leads to large gains with the Base model. An improvement of **+9** absolute points is observed for Russian. Furthermore, with the Base model, all average F1 scores are improved with the addition of synthetic data, regardless of which language the synthetic samples come from. The largest gain is seen when German samples are added **(+2.9)**.

As the size of the mT5 model increases, the gains from synthetic augmentation decrease, as shown in Tabs. 5 and 6. With the Large model, the maximum improvement in average F1 is **+1.2** absolute points. With the XL model, the average F1 scores are either the same as the zero-shot baseline or slightly lower. This is expected as when the model size increases, the gap between zero-shot and supervised also becomes smaller, hence less headroom exists when adding the synthetic samples. Fig. 5 demonstrates this scaling effect. Nonetheless, improvements of **+5.1**, **+2.2**, and **+3.4** are observed on Russian, Arabic, and Greek, respectively with the mT5 Large model. Similarly, smaller per-language gains can be seen with augmentation with the XL model, as shown in Tab. 6.

A surprising observation is that best per-language results are not necessarily achieved when augmenting with the synthetic samples from the

Dataset	en	ar	de	el	es	hi	ru	th	tr	vi	zh	avg
SQuAD en (paper)	84.6	63.8	73.8	59.6	74.8	60.3	57.8	57.6	67.9	70.7	66.1	67.0
SQuAD en (ours)	**85.5**	65.7	73.6	62.5	75.0	62.4	61.9	57.6	68.9	**71.9**	**71.1**	68.1
SQuAD en + ru	83.7	67.5	73.6	69.3	73.8	**66.2**	70.3	62.7	67.5	68.8	68.9	70.0
SQuAD en + hi	84.2	68.3	75.0	68.4	75.0	63.7	68.2	**64.5**	67.2	69.5	68.9	69.9
SQuAD en + de	84.6	**69.0**	71.8	**70.2**	**75.7**	**66.2**	**71.0**	63.5	**70.0**	70.9	**71.2**	**71.0**
SQuAD en + ar	84.5	64.0	74.4	69.4	74.4	65.1	65.1	62.5	67.9	70.0	70.2	70.2
SQuAD en + es	84.8	**69.1**	**76.1**	68.2	72.8	65.4	68.9	62.7	70.0	71.0	71.0	70.6
Supervised	83.1	72.4	76.9	76.8	79.0	71.4	76.1	67.9	72.5	75.9	76.9	75.3

Table 4: Performance of fine-tuned mT5 **Base** models on XQuAD. *Supervised* refers to training on SQuAD en + `translate-train` dataset of the target language.

Dataset	en	ar	de	el	es	hi	ru	th	tr	vi	zh	avg
SQuAD en (paper)	88.4	75.2	80.0	77.5	81.8	73.4	74.7	73.4	76.5	79.4	75.9	77.8
SQuAD en (ours)	88.6	75.0	80.4	76.5	81.6	73.9	74.1	73.8	**76.2**	**80.1**	76.4	77.4
SQuAD en + ru	88.2	76.6	81.2	79.1	82.6	76.1	77.6	72.1	75.1	78.4	77.4	**78.6**
SQuAD en + hi	**88.9**	76.7	81.5	79.4	**82.9**	73.4	78.7	**74.1**	75.0	79.1	78.0	**78.6**
SQuAD en + de	88.0	72.7	79.7	73.0	82.0	73.6	76.4	71.6	74.8	78.7	76.2	76.6
SQuAD en + ar	88.0	73.3	81.2	78.8	82.4	75.1	78.5	71.4	75.6	77.3	**78.2**	77.8
SQuAD en + es	88.2	**77.2**	**81.8**	**79.9**	81.3	**76.4**	**79.2**	72.3	75.8	79.5	77.7	**78.7**
Supervised	87.3	79.4	82.7	81.8	83.8	78.0	81.9	74.7	80.2	80.4	83.2	81.2

Table 5: Performance of fine-tuned mT5 **Large** models on XQuAD. *Supervised* refers to training on SQuAD en + `translate-train` dataset of the target language.

Dataset	en	ar	de	el	es	hi	ru	th	tr	vi	zh	avg
SQuAD en (paper)	88.8	77.4	80.4	80.4	82.7	76.1	76.2	74.2	77.7	80.5	80.5	79.5
SQuAD en (ours)	**89.7**	**79.2**	80.9	80.9	83.2	**78.7**	78.4	74.3	78.4	79.5	**80.7**	**80.2**
SQuAD en + ru	89.1	78.6	**82.1**	**81.7**	82.7	78.6	79.4	74.3	**78.7**	80.6	79.2	**80.2**
SQuAD en + hi	89.1	**79.1**	81.7	80.9	**83.4**	76.1	79.0	**74.6**	77.6	81.0	80.4	79.9
SQuAD en + de	88.8	78.2	81.2	**81.7**	82.8	78.1	**79.6**	74.0	77.7	**81.2**	79.7	80.1
SQuAD en + ar	89.0	75.0	81.3	81.5	82.8	78.4	79.3	73.5	78.4	80.2	80.4	79.6
SQuAD en + es	88.8	79.0	**82.2**	81.3	82.6	**78.7**	78.8	73.8	78.3	**81.1**	80.5	**80.2**
Supervised	88.5	80.9	83.4	83.6	84.9	79.6	82.7	78.5	82.4	82.4	83.2	82.7

Table 6: Performance of fine-tuned mT5 **XL** models on XQuAD. *Supervised* refers to training on SQuAD en + `translate-train` dataset of the target language.

same target language. Our hypothesis is that strong multilingual models such as mT5 have already developed rich per-language representations. Adding non-English synthetic data enables the model to generalize well to non-English RC tasks by not overfitting to English RC samples.

Comparing the *Supervised* metrics vs. *SQuAD en* + ⟨*lang*⟩ indicates that with Base and Large, using synthetic samples reduces the gap between the zero-shot and supervised performance of the trained RC models. This gap is reduced from **7.2** to **4.2** absolute points with the Base model. However, there still exists a sizeable gap, which could likely be further reduced through the use of higher quality synthetic samples.

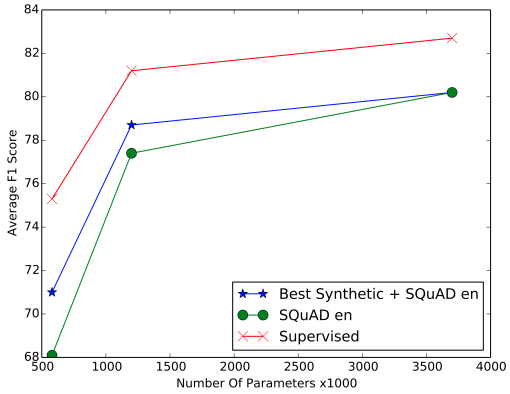


Figure 5: Scaling effect on augmentation using synthetic samples.

6 Conclusion

In this work, we presented a simple yet effective approach to generate large-scale synthetic multilingual question-answer pair data, which can be used to improve the zero-shot performance of multilingual reading comprehension (RC) models. Our experimental results showed large improvements in the performance of RC models trained on our synthetic multilingual datasets as compared to standard zero-shot baselines. Moreover, our zero-shot generation approach proved to be easily applied to any language, as long as the language is supported

by the pre-trained multilingual generative model.

While our results showed that using synthetic samples alongside English training data can significantly narrow the gap between zero-shot and supervised performance of RC models, the gap still remains. We are optimistic that future work can reduce this gap further through improved generation quality.

7 Ethical Considerations

Since the synthetic QA samples are generated by a generative model, it is possible that generated questions could include hallucinations and counterfactual information. We have employed the following safeguards: 1) The generative model is trained on the SQuAD dataset to learn question-answer generation. SQuAD is a well-studied and meticulously curated dataset. 2) The passages from which question-answer pairs are generated are selected from Wikipedia. 3) We apply round-trip filter on the generated question-answer pairs using the RC model. This approach ensures the questions are relevant to the passages. We believe these steps drastically reduce the chances of hallucinated and counterfactual samples. Nevertheless, there still exists the possibility that such bad samples could be generated. Future research efforts can explore such potential issues.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020. Reinforcement learning based graph-to-sequence model for natural question generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-Ling Mao, and Heyan Huang. 2020. Cross-lingual natural language generation via pre-training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7570–7577.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Robert M. French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Tassilo Klein and Moin Nabi. 2019. Learning to answer by learning to ask: Getting the best of GPT-2 and BERT worlds. *CoRR*, abs/1911.02365.

Vishwajeet Kumar, Nitish Joshi, Arijit Mukherjee, Ganesh Ramakrishnan, and Preethi Jyothi. 2019. Cross-lingual training for automatic question generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4863–4872, Florence, Italy. Association for Computational Linguistics.

Dong Bok Lee, Seanie Lee, Woo Tae Jeong, Donghwan Kim, and Sung Ju Hwang. 2020. Generating diverse and consistent QA pairs from contexts with information-maximizing hierarchical conditional VAEs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 208–224, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020b. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and

Luke Zettlemoyer. 2020a. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Zihan Liu, Genta Indra Winata, Andrea Madotto, and Pascale Fung. 2020b. Exploring fine-tuning techniques for pre-trained cross-lingual models via continual learning. *CoRR*, abs/2004.14218.

Xiyao Ma, Qile Zhu, Yanlin Zhou, and Xiaolin Li. 2020. Improving question generation with sentence-level semantic matching and answer position inferring. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8464–8471.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5811–5826, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Arij Riabi, Thomas Scialom, Rachel Keraron, Benoît Sagot, Djamé Seddah, and Jacopo Staiano. 2020. Synthetic data augmentation for zero-shot cross-lingual question answering. *CoRR*, abs/2010.12643.

Siamak Shakeri, Cicero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. End-to-end synthetic data generation for domain adaptation of question answering systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5445–5460, Online. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604. PMLR.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574, New Orleans, Louisiana. Association for Computational Linguistics.

Luu Anh Tuan, Darsh J. Shah, and Regina Barzilay. 2019. Capturing greater context for question generation. *CoRR*, abs/1910.10274.

Bingning Wang, Xiaochuan Wang, Ting Tao, Qi Zhang, and Jingfang Xu. 2020. Neural question generation with answer pivot. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9138–9145.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Chefbot: A Novel Framework for the Generation of Commonsense-enhanced Responses for Task-based Dialogue Systems

Carl Strathearn and Dimitra Gkatzia
Edinburgh Napier University
{c.strathearn,d.gkatzia}@napier.ac.uk

Abstract

Conversational systems aim to generate responses that are accurate, relevant and engaging, either through utilising neural end-to-end models or through slot filling. Human-to-human conversations are enhanced by not only the latest utterance of the interlocutor, but also by recalling and referring to relevant information about concepts/objects covered in the conversation so far. Such information may contain recent referred concepts, commonsense knowledge and more. A concrete scenario of such dialogues is the cooking scenario, i.e. when an artificial agent (personal assistant, robot, chatbot) and a human converse about a recipe. We will demo a novel system for commonsense enhanced response generation in the scenario of cooking, where the conversational system is able to not only provide directions for cooking step-by-step, but also display *commonsense* capabilities such as offering explanations on object use and recommending replacements of ingredients.

1 Introduction

Although conversational User Interfaces (CUIs) have gained popularity with the introduction of commercial personal assistants, these CUIs are mostly retrieval-based question answering (QA) systems that are incapable of holding a multi-turn conversation or providing follow-up information on the same topic or task. In addition, they do not incorporate commonsense capabilities, i.e. the ability to understand how an object is used, understand/infer other non-obvious properties such as its weight and materiality, or generally make inferences about ordinary tasks in our daily lives (Davis and Marcus, 2015)). Significantly, these CUIs cannot provide "how-to" instructions when performing practical tasks that require conversation over multiple steps, such as cooking a recipe or building furniture. This work is inspired by the human ability to read a document with instructions on how to perform a task and recall and rephrase the instructions to someone else. We consider a practical task where two people cook together, where one user, the Information Giver (IG) has access to the recipe and provides instructions to the second user, the Information Follower (IF), as shown in Figure 1.

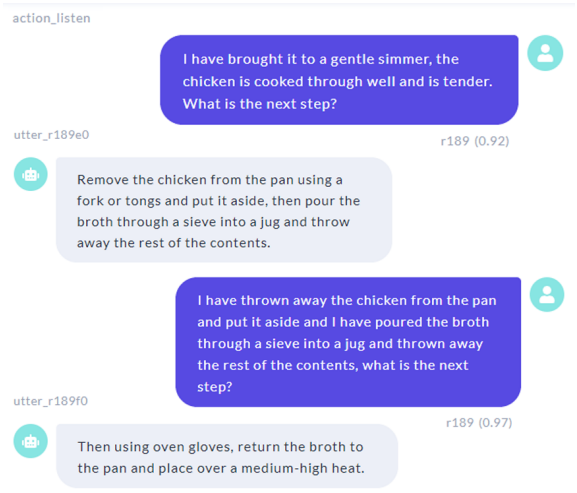


Figure 1: Excerpt from dialogue.

In our setup, the IG is required to understand the recipe, split it into manageable steps, possibly rephrase it as well as being able to describe objects, their use and common storage locations. The IF can ask questions such as repeating an instruction, clarification, and confirmation. Replicating these innate cognition processes in artificial agents that function in real-world conditions is highly challenging due to the complexity and interdisciplinary nature of the problem. Unlocking these challenges will enable artificial agents to operate with greater levels of common-sense reasoning. This demo paper presents a novel task-based CUI which combines knowledge-grounded dialogue and commonsense-enhanced response generation.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 46–47,
Aberdeen, Scotland, UK, 20-24 September 2021.

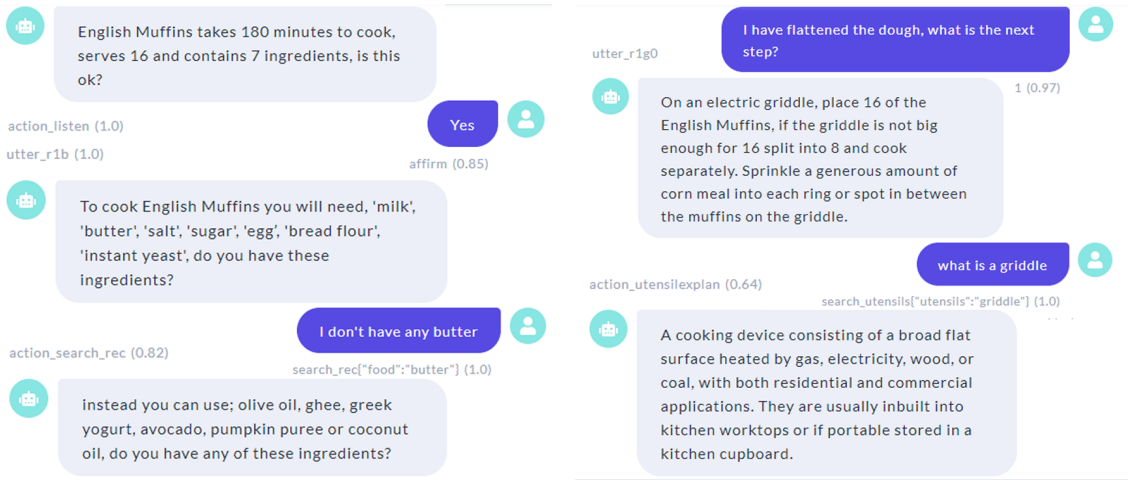


Figure 2: Examples of commonsense capabilities: On the right, Chefbot is suggesting an ingredient substitution and on the left, Chefbot explains how an object is used.

2 Task Description

Task-oriented dialogue is concerned with helping users achieve specific goals, by understanding user intends, state tracking and generating responses based on next actions (Hosseini-Asl et al., 2020). On the other hand, open-domain dialogue systems aim to converse over different topics within the same dialogue. Our proposed task is situated between these two tasks: firstly, the goal of the system is to help the user prepare a recipe by providing instructions; secondly, the system aims to converse about related concepts, such as ingredients and objects' utility in an open domain fashion (Fig. 2).

3 Chefbot

To demonstrate the use of commonsense enhanced dialogue in a practical task-based challenge, Chefbot was designed using RASA X[1]. Annotated sample conversations between the IG and IF are modelled as two modes of question and answer pairs. The first series of utterances are open-domain and the second set are domain specific. In the dialog flow, forms were used for each recipe to force the sequence between the two series allowing for both domain and non-domain utterance classification. This produced a more robust structure and contextual awareness for state tracking and response generation. The Chefbot is able to handle questions that are not represented in the sample dataset with the help of two commonsense databases. The first database provides the user with appropriate alternative ingredients for a specific recipe and the second explains the use, handling, alternative names and typical storage locations of kitchen utensils. A combination of rules, checkpoints and custom actions, allow the user to ask questions at any stage in the task and then on fulfilment, return the the next logical step in the recipe. From this framework we create a multi-intent / multi-turn policy model that permits adaptability to cope with the variable conditions of real-world tasks.

4 Future Work

In future, we aim to extend our system so it can be used in a situated Human-Robot Interaction scenario, where the conversation will take place as a spoken conversational interaction.

5 Conclusions

This demo paper describes a commonsense-enhanced chatbot for task-based dialogue. At INLG, we will demo the chatbot and will discuss initial findings.

Acknowledgements

The research is supported under the EPSRC projects EP/T014598/1 and EP/T024917/1.

References

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Commun. ACM*, 58(9):92–103.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191.

[1] https://rasa.com/docs/rasa-x/

Predicting Antonyms in Context using BERT

Ayana Niwa [†], **Keisuke Nishiguchi**[‡], **Naoaki Okazaki**[†]

[†] Tokyo Institute of Technology
[‡] CyberAgent, Inc.
ayana.niwa at nlp.c.titech.ac.jp
nishiguchi_keisuke at cyberagent.co.jp
okazaki at c.titech.ac.jp

Abstract

We address the task of antonym prediction in a context, which is a fill-in-the-blanks problem. This task setting is unique and practical because it requires contrastiveness to the other word and naturalness as a text in filling a blank. We propose methods for fine-tuning pre-trained masked language models (BERT) for context-aware antonym prediction. The experimental results show that these methods have positive impacts on the prediction of antonyms within a context. Moreover, human evaluation reveals that more than 85% of the predictions using the proposed method are acceptable as antonyms.

1 Introduction

Antonymy is a relationship between two words that express contrasting or opposite meanings (e.g., "agree–disagree"). Capturing antonymy is directly helpful for downstream applications such as sentiment transfer (Li et al., 2018) and claim generation (Hidey and McKeown, 2019). Further, semantically contrasting expressions with antonyms are utilized in advertising slogans (Katrandjiev et al., 2016), political speeches (Heritage and Greatbatch, 1986), and Chinese poetry (Yan et al., 2016).

As antonymy is one of the relations of lexical semantics, such as synonymy and hyponymy, antonymy can be modeled using a similar approach to lexical knowledge acquisition. Most of the published studies on this topic have focused on the prediction of the relation between a given word pair (Barkan et al., 2020; Shwartz and Dagan, 2016), or a target (tail) for a given word (head) and its relation (Camacho-Collados et al., 2018; Rimell et al., 2017). However, predicting an antonym is challenging because multiple types of words are plausible as antonyms for a word. This is because a word can have semantic contrastiveness to the other as long as the word contains at least one feature contrasting to the other (Leech, 1976). For example, *dual*, *double*, and *multiple* can be antonyms for *single* because they all have the contrasting features of AMOUNT or NUMBER. Additionally, the appropriateness of an antonym varies depending on the context. For example, *double*, *dual*, and *multiple* are used for a bed, nationality, and the number of meanings of a word, respectively. Hence, antonym prediction must be considered within the context.

In this study, we consider the new task of antonym prediction, that is, **the fill-in-the-blanks problem for antonyms in context**. For example, in the sentence, "A ____ bed is better than *single* for me," we expect to fill the blank with the words "*double*" or "*king-sized*." The fill-in-the-blanks setting requires the prediction of context-aware antonyms by capturing the contrasting features between the word pair. The task also requires a consideration of the naturalness of a text when filling the blank, which is necessary for applications of generating text with antonyms.

In recent years, pre-training and fine-tuning approaches have achieved high performance in various NLP tasks (Devlin et al., 2019; Yang et al., 2019). Therefore, we use Bidirectional Encoder Representations from Transformers (BERT) as a pre-trained model to predict antonyms in a context. However, it is not easy to collect training data for fine-tuning the model, that is, text containing antonym pairs with a contrastive context.

Therefore, we focus on the rhetorical device that effectively employs antonymy, that is, *antithesis*, which juxtaposes words or phrases in a similar structure with contrasting meanings. An antithesis is suitable for data creation because it ensures that a text has one or more antonym pairs in a contrastive context. For example, the sentence, "My mother who [is *sensitive* to the *pension*] [is *insensitive* to the *insurance*]," has an antithesis structure with two antonym pairs. We propose four methods to fine-tune BERT for antonyms: (1) domain adaptation using an antithesis corpus, (2) contrastive masking to focus on antonym prediction, (3) antithesis po-

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 48–54,
Aberdeen, Scotland, UK, 20-24 September 2021.

sitional encodings to capture antithesis structures, and (4) pseudo-supervision data collected by automatic annotation using an antonym dictionary.

The experimental results with the Japanese slogan corpus demonstrate that the proposed fine-tuning methods contribute to the adaptation of BERT to the context-aware antonym prediction task. An automatic evaluation based on a single correct answer is improper because there are multiple acceptable answers. However, the manual evaluation revealed that more than 85% of the words predicted by the proposed method are appropriate as antonyms and that fine-tuned BERT is highly capable of capturing antonymy in a context.

2 Method

2.1 Model

Given a sequence of n tokens, $x_1, \ldots, x_n$, with a `[MASK]` (blank) token at position m ($1 \leq m \leq n$), the conditional probability of token y_m for filling the blank can be modeled using a BERT (Devlin et al., 2019) (illustrated in Figure 1),

$$P(y_m | x_1, \ldots, x_m, \ldots, x_n). \quad (1)$$

Based on the bidirectional contexts of the input, BERT considers the surrounding context of `[MASK]`. By fine-tuning BERT on a text corpus with an antithesis structure (described in Section 2.2), we can expect that the model will eventually consider antonymy in an input text by domain adaptation because an antithesis contains more than one antonym pair and the contrastive context.

To utilize a small corpus for adapting BERT for antonyms, we explore two approaches. First, we create supervision data for fine-tuning by replacing a token with `[MASK]` such that the `[MASK]` token is likely to have a counterpart in the text. For example, given a text, "[Starts with the reckoning], [ends with the relish]," we obtain two training instances, that is, "`[MASK]` with the `[MASK]`, ends with the relish" and "Starts with the reckoning, `[MASK]` with the `[MASK]`." These `[MASK]` tokens are chosen because they do not appear in the counterpart phrase, whereas "with" and "the" do. We refer to this strategy as *contrastive masking*. This strategy selectively creates supervision data for filling antonyms more efficiently than the default strategy for BERT (deciding `[MASK]` positions randomly).

Second, we extend the positional encodings in BERT to indicate an antithesis structure in an input. Consider a text that includes two spans $[i, j)$ and $[k, l)$ ($1 \leq i < j \leq k < l \leq n$) forming an antithesis structure and the span $[i, j)$ includes `[MASK]` tokens. To indicate that span $[i, j)$ corresponds to $[k, l)$, we compute an index specialized for the antithesis structure,

$$a_t = \begin{cases} k + \left\lfloor \frac{(l-k)(t-i)}{j-i} \right\rfloor & (a \in [i, j)) \\ t & (\text{otherwise}) \end{cases}. \quad (2)$$

The index a_t represents the position of the token x_t if $t \notin [i, j)$ but that of the counterpart span $[k, l)$ if $t \in [i, j)$. We used the mean of absolute positional encoding (used in the original BERT) and *antithesis positional encodings* (indexed by a_t) as the positional encodings for BERT.

2.2 Supervision data

For the domain of the training data, we selected advertising slogans in which antitheses were likely to be used frequently. We used a corpus of advertising slogans that consisted of 111,295 Japanese slogans collected from existing books (Taniyama, 2007; Nakahata, 2008; Aota et al., 2007; Umeda, 2016; Sendenkaigi Award Committee, 2003–2018) to construct the supervision data. With the slogans, we constructed an antithesis corpus manually by crowd-sourcing two subtasks: (1) filtering out slogans that do not contain antithesis structures with strict criteria, and (2) annotating antithesis spans. This process yielded 7,457 slogans with annotated spans of antitheses[1]. Additional information is provided in the Appendix.

2.3 Pseudo-supervision data

The number of instances in the supervision data may be small for fine-tuning BERT. Thus, we also explore an approach for automatically annotating a text in a manner similar to distant supervision. Specifically, we find slogans that include pairs of antonyms included in the antonym dictionary (Sanseido Editorial Office, 2017). This process resulted in 1,894 slogans that were not included in the dataset explained in Section 2.2. The strict criteria filtered out these slogans in the first step of the corpus construction, but some of them actually presented antitheses. For each pair of antonyms in a slogan, we obtain two training instances, one antonym replaced with the `[MASK]` token, and vice versa. In this way, we inject the lexical knowledge of antonyms into BERT.

[1] Unfortunately, we cannot release the corpus to the public because we do not own the copyrights of the slogans.

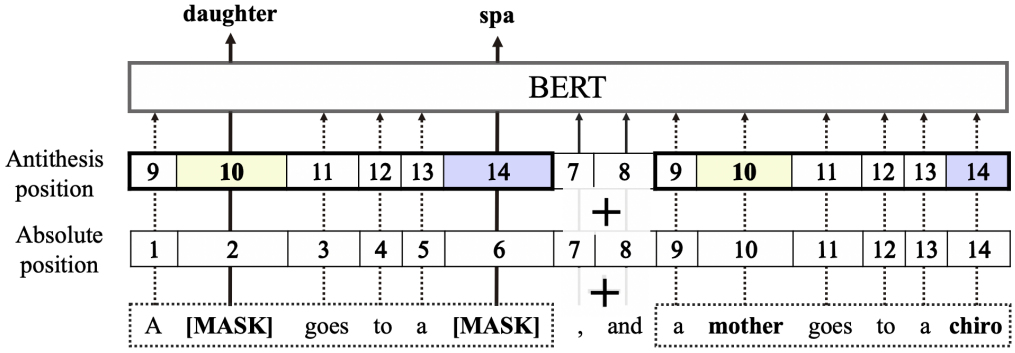


Figure 1: Outline of the proposed method. This is an example of predicting "daughter" and "spa" for the input text "A [MASK] goes to a [MASK], and a mother goes to a chiro." The phrases, "A [MASK] goes to a [MASK]" and "a mother goes to a chiro" consist of an antithesis structure.

3 Experiments and Results

3.1 Experimental settings

Dataset We split the 7,457 slogans in the antithesis corpus into training, development, and test data, and subsequently converted them into fill-in-the-blank instances that contained the [MASK] tokens. Each text yielded two masked instances because an antithesis structure has two contrastive phrases. In this manner, we obtained 11,922 training, 1,496 development, and 1,496 test instances. We also used 3,788 training instances from the pseudo-supervision data. In addition to them, we created a subset of the test data ("word level" hereafter) for a fair comparison with the human baseline. The above test data contained instances wherein the [MASK] token is split into subword units, which human subjects cannot fill in. Additionally, our fill-in-the-blank problem is complex for general cloud workers to solve because it require an understanding of the context. Therefore, we created the simple "word level" test set based on the following criteria: (1) a single word is selected as a masked token per test instance, (2) the selected word is not split into subwords, and (3) the part of speech is either a noun, verb, adjective, or adjective-verb. Furthermore, we randomly selected only one test instance per slogan to simplify the crowd-sourcing process. This process created 529 word level test instances, which is smaller than the entire test set.

Baselines We used two baselines, dictionary-lookup and pre-trained BERT without fine-tuning. The dictionary-lookup baseline examines whether the gold word of each blank is registered in the dictionary (Sanseido Editorial Office, 2017) as an antonym of any word in the corresponding phrase[2]. A pre-trained BERT without fine-tuning is evaluated to investigate the ability of the model to predict antonyms in a context without specialized training. We used BERT pre-trained with the Japanese Wikipedia[3]. To assess the difficulty of this task, we asked three human subjects to guess at most five possible words to fill the blanks in the test set.

We do not employ a non-contextual baseline other than the dictionary-lookup because the dataset has annotations of antithesis structures only at the segment level (phrase-to-phrase alignment) but not at the word level (word-to-word alignment).

Evaluation metrics We used top-1 and top-10 accuracy values as the measures for the correctness of model predictions. Because human subjects could not always come up with five answers for a blank, we used the top-1 and top-n accuracy values, wherein the number of n varied depending on the number of human responses in each instance.

3.2 Results

Table 1 reports the accuracy of the prediction of blanks on the test data. The proposed method achieved 29.3% top-1 and 53.8% top-10 accuracies measured for all instances, and 30.4% top-1 and 49.1% top-n accuracies measured at the word level. The pre-trained BERT without fine-tuning obtained much lower accuracies than those of the proposed method. This indicates that a general masked language model was insufficient to predict antonyms even if presented in the context of the antithesis structure.

[2]This baseline presents the upper bound of the performance of dictionary-lookup because it knows the gold words.

[3]https://github.com/cl-tohoku/bert-japanese

	All		Word level	
	Acc@1	Acc@10	Acc@1	Acc@n
dictionary-lookup	-	-	9.6	-
pre-trained BERT (w/o fine-tuning)	15.0	40.9	15.7	39.1
fine-tuned BERT (default masking)	24.4	51.4	25.0	44.4
- default masking + contrastive masking	28.8	52.6	27.4	47.4
+ antithesis positional encodings	28.7	53.5	27.4	48.0
+ pseudo-supervision data	**29.3**	**53.8**	**30.4**	**49.1**
human (lowest)	-	-	31.5	52.3
human (highest)	-	-	34.5	59.1
human (votes from three subjects)	-	-	51.8	66.6

Table 1: Accuracy values of antonym prediction.

	Contrastiveness	Naturalness
human	94	90
method	88	85

Table 2: Number of contrastive and natural instances (out of 100) judged by a human.

Fine-tuning BERT on the supervision data boosted the performance, especially for top-1 predictions (+9.4 and +9.3 points). Contrastive masking improved all the accuracies, and antithesis positional encodings improved the top-10 and top-n accuracies in particular (+1.2 to +3.0 points for the former and +0.9 and +0.6 points for the latter). Moreover, we confirmed that the pseudo-supervision data improved the accuracy, especially for top-1 predictions (+0.6 and +3.0 points). The fact that these proposed methods contribute to the performance shows the importance of fine-tuning BERT with a special focus on antonym prediction.

The baseline of dictionary-lookup obtained 9.6% top-1 accuracy measured at the word level. We found that the low coverage of the dictionary was the leading cause: the dictionary had entries for only 39.3% of antonyms in the test data.

Table 1 also illustrates the results of human subjects who had the lowest and highest accuracy when solving the fill-in-the-blank task. The accuracy values of all the human subjects were better than those of the proposed method, although the performance of each human subject varied. However, even the best-performing human subject could achieve 34.5% top-1 and 59.1% top-n accuracies, which justifies the difficulty of this task. Conversely, with the most lenient evaluation in which we regard a prediction as correct if any of the three human subjects provided the right answer, the top-1 accuracy was 51.8%, and the top-n accuracy was 66.6%. The performance increase in top-1 implies that multiple words are acceptable for the blanks in the test data, and the characteristic is considered the reason why even human subjects cannot achieve high accuracy values. To investigate such cases with multiple possible correct words, we conducted a subjective evaluation of the quality of the answers of the human subjects and the proposed method (with the participation of another human subject). For this analysis, we used 100 instances sampled at random from the test data for which the answers of both human subjects and methods did not match the correct word. We chose an answer from three subjects at random for each instance because we had multiple answers from three subjects. Table 2 reports the number of predictions from the human subjects and the proposed method for which the manual evaluation recognized contrastiveness and naturalness (fluency). The results reveal that more than 85% of both the answers of the human subjects and the predicted words of the proposed method are appropriate as antonyms.

To summarize, we found that the automatic evaluation using a single correct answer (accuracy) underestimated the context-aware antonym prediction performance because there could be multiple acceptable answers. However, the subjective evaluation revealed that the predictions of the proposed method were satisfactory in terms of contrastiveness (as an antonym) and naturalness in a context.

3.3 Analysis

We list the predictions by the baseline (BERT without fine-tuning) and the proposed method, and the answers by each human subject in Table 3.

When the antonymy was easy to understand,

Example (A)		Example (B)	
別れの曲だったのに、[MASK]の曲になった。 It was the farewell song, but became the [MASK] song.		地球の環境より、まず[MASK]の環境。 Put the environment of the [MASK], before the environment of the earth.	
correct answer: 出会い encounter		correct answer: 心 mind	
baseline	別れ, 最後, 今, 人生 farewell, last, present, life's	baseline	宇宙, 水, 地球, 太陽, 植物 universe, water, earth, sun, plants
proposed	出会い, 憧れ, 最高, 始まり encounter, longing, best, beginning	proposed	家族, 私, 周り, トイレ, 家 family, of myself, vicinity, restroom, house
human 1	出会い, 再会, 初恋, 永遠 encounter, reunion, first love, eternal	human 1	自宅, 自分, 部屋, 職場 one's home, of myself, room, workplace
human 2	出会い, 始まり, 邂逅 (unexpected) encounter, beginning	human 2	自分, 私, 周辺, 室内, 家内 of myself, vicinity, room, one's wife
human 3	出会い encounter	human 3	国, 家庭, 町, 周り country, family, town, vicinity

Table 3: Examples of prediction of methods and answers of human subjects. Owing to space limitations, we removed some duplicated words, which are synonyms (e.g., beginning and start) or the same word in different character types (e.g., Hiragana and Kanji in Japanese).

such as "farewell–encounter" in Example (A), both the proposed method and the human subjects could output the correct word as the first candidate. Compared to the baseline, the proposed method could focus on the contrastiveness between the phrases "the farewell song–the encounter song."

In Example (B), the correct word was not predicted or answered, but their outputs were the antonyms. The output word should be contrasted with the word "earth" in terms of its scale and degree of familiarity. In this respect, both the prediction of the proposed method and the answers of the human subjects satisfied the semantic contrast and naturalness as sentences because they were lined up with words that mainly referred to objects and people around them, and all of them satisfied the semantic contrast and naturalness as sentences. However, the correct answer was "mind," which has "physical and mental contrasts" in addition to perspectives of scale and familiarity. To deal with these cases, it is necessary to clarify from what perspective the two words are contrasted.

Some cases were difficult to predict both by the proposed method and human subjects. Such instances require prior knowledge and imaginations about objects mentioned in the text (advertisement targets in case of slogans), for example, "From lightness within [MASK] to lightness almost weightless," where the gold answer is "tolerance" for a *glass* product. It requires additional input information about the target of the sentence to deal with such cases.

4 Conclusion

In this study, we addressed the task of predicting antonyms within a context. We proposed methods for adapting BERT to antonym prediction, such as domain adaptation using an antithesis corpus, contrastive masking, antithesis positional encodings, and pseudo-supervision data collection. The proposed method achieved 29.3% top-1 and 53.8% top-10 accuracies on the test data. Although these values seem low, an automatic evaluation based on a single correct word underestimates the performance because multiple valid words can fill in the blanks. The subjective evaluation revealed that more than 85% of the words predicted by the proposed method were appropriate as antonyms. Our proposed task and method will be useful in many real-world applications that use contrastive expressions. Although we used Japanese text in this study, it can be applied to any language as far as the annotated data is available. In the future, we will extend the proposed method to generate text with antithesis, and explore the fill-in-the-blanks problem setting for other semantic relations.

References

Mitsuaki Aota, Akira Akiyama, Hideki Azuma, et al. 2007. *Saishinyaku copy bible (in Japanese) (English translation:The brand new slogan bible)*. Sendenkaigi Co., Ltd.

Oren Barkan, Avi Caciularu, and Ido Dagan. 2020. Within-between lexical relation classification. In

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 3521–3527, Online. Association for Computational Linguistics.

Jose Camacho-Collados, Claudio Delli Bovi, Luis Espinosa-Anke, Sergio Oramas, Tommaso Pasini, Enrico Santus, Vered Shwartz, Roberto Navigli, and Horacio Saggion. 2018. SemEval-2018 task 9: Hypernym discovery. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 712–724, New Orleans, Louisiana. Association for Computational Linguistics.

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

John Heritage and David Greatbatch. 1986. Generating applause: A study of rhetoric and response at party political conferences. *American journal of sociology*, 92(1):110–157.

Christopher Hidey and Kathy McKeown. 2019. Fixed that for you: Generating contrastive claims with semantic edits. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1756–1767, Minneapolis, Minnesota. Association for Computational Linguistics.

Hristo Katrandjiev, Ivo Velinov, and Kalina Radova. 2016. Usage of rhetorical figures in advertising slogans. *Trakia Journal of Sciences*, 14(03):267–274.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics.

Geoffrey Leech. 1976. *Semantics*. Penguin Books.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Takashi Nakahata. 2008. *Honto no koto wo iu to, yoku, shikarareru. katsu copy no zenbu (in Japanese) (English translation : We are often scolded when we say the truth. All of the slogans for winning.)*. Sendenkaigi Co., Ltd.

Laura Rimell, Amandla Mabona, Luana Bulat, and Douwe Kiela. 2017. Learning to negate adjectives with bilinear models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 71–78.

Sanseido Editorial Office, editor. 2017. *Hantaigo Tairitsugo Dictionary (in Japanese) (English translation : Antonym Dictionary)*. Sanseido Co.,Ltd.

Sendenkaigi Award Committee. 2003–2018. *SKAT.2–SKAT.17*. Sendenkaigi Co., Ltd.

Vered Shwartz and Ido Dagan. 2016. Path-based vs. distributional information in recognizing lexical semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 24–29, Osaka, Japan. The COLING 2016 Organizing Committee.

Masakazu Taniyama. 2007. *Koukoku copy tte kou kakunnda! dokuhon (in Japanese) (English translation : This is how you write advertising slogans! A textbook)*. Sendenkaigi Co., Ltd.

Satoshi Umeda. 2016. *"Kotoba ni dekiru" ha buki ni naru. (in Japanese) (English translation:The ability to "put into words" is a weapon.)*. Nikkei Publishing Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Rui Yan, Cheng-Te Li, Xiaohua Hu, and Ming Zhang. 2016. Chinese couplet generation with neural network structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2347–2357.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

A Construction of an antithesis corpus

As described in Section 2.2, the annotation process for constructing the antithesis corpus was divided into the following two steps: (1) identification of candidate antitheses and (2) annotation of the span of the antithesis. In step (1), we assigned each slogan to five workers to determine whether the slogan contained an antithesis. If more than three workers determined that the slogan contained an antithesis, we would consider it as a candidate antithesis. Thus, we succeeded in extracting 9,720 slogans that contained antitheses. In step (2), we selected two workers with high-annotation quality and asked them to annotate each antithesis with its span, for example, "[A lean body] leads [a bold life]."

B Model architectures and implementation details

We used $\mathrm{BERT}_{\mathrm{BASE}}$, which has 12 layers, 768 hidden states, 12 heads, and 110M parameters for all the experiments. During the pre-training, the whole word masking was enabled. We used Mecab (Kudo et al., 2004) as the tokenizer.

Our implementation, including the code for evaluation, was based on Huggingface Transformers (Wolf et al., 2020). In fine-tuning BERT with the AdamW (Loshchilov and Hutter, 2019) optimizer, we set a batch size of 8, a maximum sequence length of 50, and the remaining parameters were set to the default values. The experiments were run on servers with an Nvidia Tesla P100 GPU. The total number of epochs for the fine-tuning was 6, determined by the accuracy of development data.

Examining Covert Gender Bias: A Case Study in Turkish and English Machine Translation Models

Chloe Ciora*, **Nur Iren***, **Malihe Alikhani**
University of Pittsburgh
{chloeciora,nei3,malihe}@pitt.edu

Abstract

As Machine Translation (MT) has become increasingly more powerful, accessible, and widespread, the potential for the perpetuation of bias has grown alongside its advances. While overt indicators of bias have been studied in machine translation, we argue that covert biases expose a problem that is further entrenched. Through the use of the gender-neutral language Turkish and the gendered language English, we examine cases of both overt and covert gender bias in MT models. Specifically, we introduce a method to investigate asymmetrical gender markings. We also assess bias in the attribution of personhood and examine occupational and personality stereotypes through overt bias indicators in MT models. Our work explores a deeper layer of bias in MT models and demonstrates the continued need for language-specific, interdisciplinary methodology in MT model development.

1 Introduction

Various forms of biases are encoded in the way that people use language (Rudinger et al., 2018; Butler, 1990). Similar to other Natural Language Processing (NLP) tasks, learned models used in MT systems include social biases as they learn correlations from their training data that have encoded stereotypes. Specifically, several studies (Prates et al., 2020; Cho et al., 2019; Baeza-Yates, 2019) have shown that translations from a gender-neutral language to a language with gendered pronouns are biased in the selection of pronouns in the target language.

However, this is not the only way bias can manifest in MT. For example, Figure 1 demonstrates marked gender in the female case of the same sentence while remaining neutral in the male case. Since the translations are both accurate, unless the two sentences are presented together, the asymmetry in gender reference is not immediately obvious. The example demonstrates the use of optional referential gender in Turkish, highlighting the need to frame gender bias in MT around language-specific social and cultural knowledge.

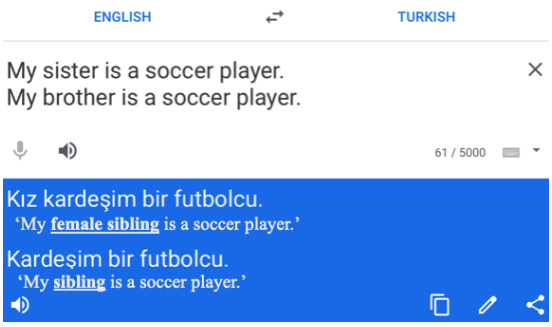


Figure 1: Using Google Translate, "My sister is a soccer player" accurately translates to "My **female sibling** is a soccer player" while "My brother is a soccer player" is translated to "My **sibling** is a soccer player". Gender is overtly marked only when the subject is female.

While previous mitigation efforts have focused on debiasing training data (Elaraby et al., 2018; Costa-jussà and de Jorge, 2020; Stafanovičs et al., 2020; Saunders and Byrne, 2020), the issue of covert bias has not been adequately addressed, and goes far beyond the perpetuation of outdated stereotypes. In order to ensure that the true meaning of the source is accurately represented during the translation process, understanding the linguistic and social context of the utterance is necessary.

In this paper, we examine both overt and covert gender biases in commercially-used MT models through the use of a gender-neutral language, Turkish, and a gendered language, English. Our study investigates explicit stereotype bias through the assignment of pronouns according to stereotypes regarding occupation and personality. We also investigate how additional qualifiers to job descriptions affect results: for example, are "good doctors"

* Equal contribution.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 55–63,
Aberdeen, Scotland, UK, 20-24 September 2021.

more likely to be men than "bad" ones? Similarly, we measure how a reference to personhood changes pronoun results. Lastly, we shed light on the presence of asymmetrical gender in MT models by analyzing explicit gender markings in Turkish translations of gender-specific English sentences. We not only ask if gender markings occur more for female subjects, but also if gender markings are more likely when the stereotype of the predicate does not align with the gender of the subject.

To this end, we created a parallel corpus of 1,617 Turkish and English job titles. We also compiled a list of descriptive adjectives based on Turkish stereotypes and formed appropriate Turkish sentences with and without a reference to personhood. Lastly, we formed a dataset of English sentences by pairing a gendered English subject word (that has no gendered translation in Turkish) with a gender-stereotyped action or description. Our code and data can be found in our GitHub repository.[1]

2 Related Work

Previous works on bias in embeddings and models (Bolukbasi et al., 2016; Zhao et al., 2019; Stanovsky et al., 2019), as well as corpora (Babaeianjelodar et al., 2020), have demonstrated that gender bias exists in the core of MT models. Additionally, Stanovsky et al. (2019) introduced a challenge set in measuring bias from English to languages with morphological gender.

One common approach in bias evaluation is to translate from a gender-neutral language to a gendered language and examine the pronouns selected for occupations and adjectives (Prates et al., 2020; Farkas and Németh, 2020; Cho et al., 2019). We used a modified version of these methods by ensuring that the occupation exists in the target language as well as the source language and that the adjectives used are actual stereotypes in Turkey (Sakallı et al., 2018)[2]. Our remaining experiments are inspired by socio-linguistics research in Turkish. First, Braun (2001) discusses how neutral Turkish words describing people, such as *insan* ("human"), tend to be biased towards male interpretations. In NLP, Mehrabi et al. (2020) examines a related bias in English named-entity recognition where fewer female names are recognized as "person" entities than male ones. Our work will similarly examine gender and personhood bias but in MT models. Second, Braun (2001) describes asymmetrical gender markings in the Turkish language, concluding that male gender remains unmarked regardless of context, whereas female gender tends to be overly expressed. For example, female children are more likely to be referred to with marked gender (*kız çocuğu* "girl child" instead of *çocuk* "child") than male children. The exception to this pattern is when the subject is exceptionally stereotyped as feminine (e.g. *hizmetçi* "househelper"). We will extend the study of this phenomenon to MT.

3 Experiments

We used four commercially available MT models in our experiments: Google Translate, Amazon Translate, Microsoft Translator, and SYSTRAN. For reproducibility purposes, all translations were executed in April of 2021. All datasets can be found on our GitHub[1].

3.1 He is a Doctor, She is a Nurse? Gender Bias in Job Occupation

We examined the distributions of the pronouns selected in English when Turkish sentences were translated following the template[3]: "He/She is a(n) <occupation>", and compared them to the 2020 Turkish (Türkiye Istatistik Kurumu, 2021) and US (U.S. Bureau of Labor, 2020) workforce statistics. Inspired by Farkas and Németh (2020), a second template "He/She is a <adjective> <occupation>" was also formed using the words *çok kötü* ("very bad"), *kötü* ("bad"), *iyi* ("good"), and *çok iyi* ("very good") as attributive adjectives to determine their influence.

We retrieved occupation lists from Turkish and US government agencies[4] and matched occupations that exist in both countries [5]. Some occupation titles were modified for clarity, and some were removed due to gender requirements or a lack of census data, as described in detail in Appendix A. Through our matching process, we were able to match 1,617 occupations.

[1] https://github.com/NurIren/Gender-Bias-in-TR-to-EN-MT-Models

[2] Turkish is also a commonly used gender-neutral language in previous works (Prates et al., 2020; Lauscher and Glavaš, 2019; Zhao et al., 2020), but these works use an intermediary translator to form their Turkish datasets.

[3] The same template was also used by Prates et al. (2020).

[4] Turkish Employment Agency (İŞKUR) and the United States Department of Labor Bureau of Labor Statistics

[5] Using both the major and minor occupational titles of International Standard Classification of Occupations (ISCO-08)

3.2 He is Smart, She is Beautiful? Bias in Adjective Use

We pulled stereotypes from a study where Turkish undergraduate students were asked to provide adjectives that describe men and women (Sakallı et al., 2018). We compiled the list of adjectives presented by this work and removed any that were lexically gendered, leaving 97 total adjectives. Each adjective was then labeled as either masculine-coded (e.g. *agresif* "agressive") or feminine-coded (e.g. *güçsüz* "weak") if more than 60% of the time that word was used to describe a certain gender. All others were considered to be neutral.

The adjectives were first placed into the template "O <adjective>" (He/She is <adjective>)[6] to assess the adjective stereotypes and then into the template "O <adjective> birisidir" ("He/She is someone who is <adjective>)[7] in order to assess if the introduction of the "personhood factor" changed the assumed gender in the translations.

[6]Since Turkish is an agglutinative language, the proper suffixes were also appended to each adjective in order to fit the first template.

[7]Note that although the translation may seem unnatural in English, this is a common utterance in Turkish.

3.3 Bias Through Asymmetrical Gender Markings

English sentences were formed with grammatically gendered subjects, followed by a predicate including a stereotypical occupation, description, or activity. For example, "My sister is an engineer" contains a female subject and a stereotypically masculine predicate. These sentences were then translated to Turkish to measure if the subject was gender-marked. We aim to answer several questions. First, are sentences with male subjects less likely to mark gender than sentences with female subjects? Second, is gender more likely to be marked when the stereotype of the predicate does not align with the gender of the subject?

We selected four subject words that are gendered in English but are grammatically neutral in Turkish. For example, there are no commonly used words for "brother" and "sister"; the only options are "sibling" (*kardeş*), "male sibling" (*erkek kardeş*), or "female sibling" (*kız kardeş*). For each of the predicate categories (occupation, description, and activity), we selected five that were stereotypically masculine and five stereotypically feminine according to Turkish gender stereotypes (Sakallı et al.,

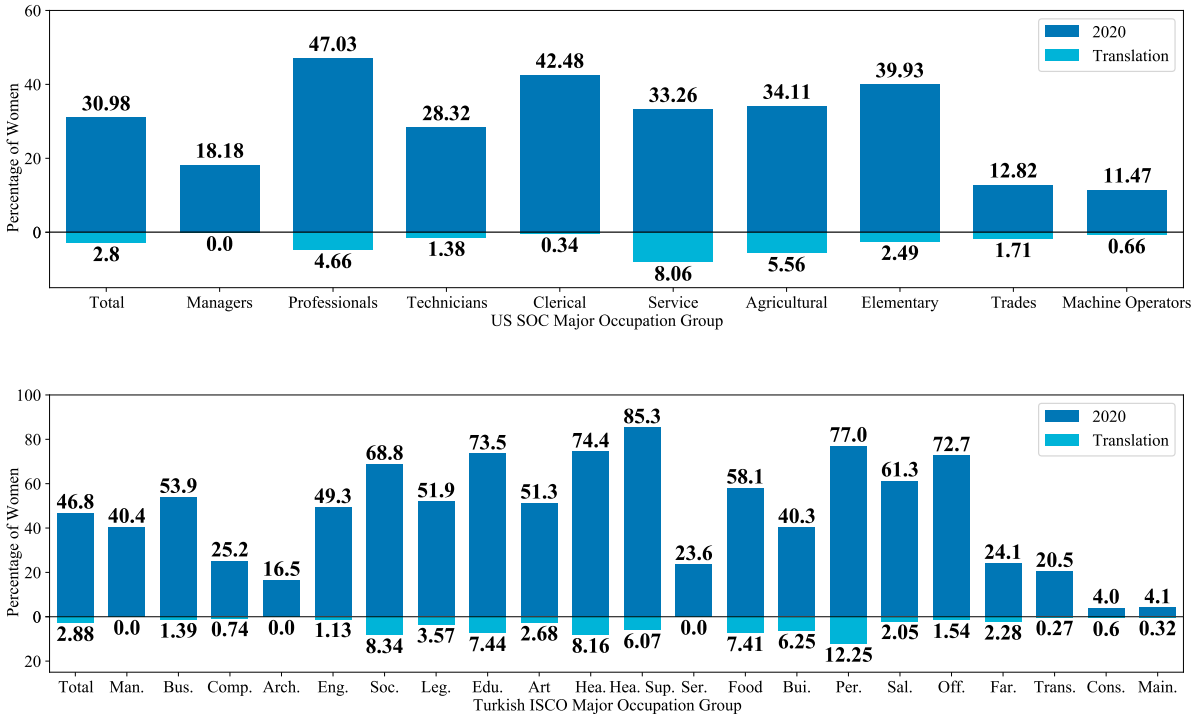


Figure 2: Comparison of the percent of women in the Turkish (bottom) and US (top) labor force in 2020 with the average of the MT models broken down by ISCO-08 and SOC major groups. In both breakdowns, the translation results clearly do not match the labor force. Additionally, the percent of female translations tends to increase in ISCO groups (bottom) with higher female participation. Full group names can be found in Appendix A.

2018; Vatandaş, 2011).

By checking the translations for overt gender markings, translators can be evaluated for asymmetry. We compared the results across the gender of each original English subject word as well as the stereotypical gender of each predicate. With 10 sentence templates in each category for the four gendered subject words, we constructed 120 sentences for each gender in total.

4 Results

In this section, we evaluate[8] aggregate results across all experiments.

4.1 Gender Bias in Occupations

Overall, the percent of female pronouns selected by the MT models were: 1.11% with Google, 1.18% with Amazon, 3.83% with Microsoft, and 5.07% with Systran. Figure 2 demonstrates that this is drastically low compared to female participation in the 2020 workplace in Turkey (31.78%) and the US (47%).

The SOC 2018 group breakdown reveals that, for occupation groups where female participation is either approximately equivalent to or greater than male participation, the models tended to translate the occasional occupation with a female pronoun. Occupations where women are the minority tended to have none or nearly no female translations. Additionally, stereotypical occupations like nurses, fashion designers, and beauticians[9] were consistently translated with female pronouns. Overall, assuming the translation results in each job category should match the corresponding labour statistic, our results were statistically significant ($p < 0.01$).

4.2 Impact of an Attributive Adjective Preceded by Occupation

As shown in Table 1, when an adjective was introduced, sentences originally assigned a female pronoun were more likely to be assigned a male pronoun instead. For each attributive adjective, this was statistically significant ($p < 0.01$). Furthermore, as the adjective changed from *çok iyi* "very good" to *çok kötü* "very bad", the amount of female pronouns that changed to male increased, but the reverse occurred for male pronouns. For example, using Google, Amazon, and SYSTRAN, the Turkish sentence "*O bir Yoğun Bakım Hemşiresi*" yielded the translation "She is an intensive care unit nurse", but the sentence "*O çok kötü bir Yoğun Bakım Hemşiresi*" yielded "He is a very bad intensive care unit nurse".

Adjective	"She"→"He"	"He"→"She"
Very Good	**0.1272**	0.0044
Good	**0.1503**	0.0039
Bad	**0.3353**	0.0005
Very Bad	**0.3815**	0.0010

Table 1: The proportion of pronouns that changed (female to male or male to female) due to the addition of an attributive adjective, cumulative across all translators.

4.3 Turkish Gender Stereotypes in Person Descriptors

For the first sentence template ("He/She is <adjective>"), the first outstanding result is that only 6.74% percent of the pronouns assigned were female (SYSTRAN: 24.5%, Google: 2%, Microsoft: 3.1%, Amazon: 2%) which indicates a strong bias towards male pronouns overall. Secondly, the sentences that were translated to a female pronoun were much more likely to have contained a female-coded adjective. This was highly significant ($p < .01$) in comparison to the amount of female pronouns generated by sentences with male-coded adjectives and significant ($p < .05$) in comparison to neutral ones. The reverse did not hold true for male pronouns; while 83.34% of all sentences that were assigned a female pronoun contained female-coded adjective, only 46.70% of translations with male pronouns were male-coded.

4.4 Analyzing Gendered Personhood

Following from the previous section, we analyze if adding a personhood modifier to the adjective sentences affects pronoun use. Of the sentences that were assigned female gender in the first template, 74.07% changed to male pronouns in the second template when personhood was introduced. The opposite is not the case; only 2.76% of adjectives with male pronouns in Template 1 were female in Template 2. Overall, each translator was significantly more likely to assign a male pronoun when the original sentence contained a personhood modifier ($p < 0.01$).

4.5 Asymmetrical Gender Analysis

As shown in Figure 3, for male subject words, 47.7% of the translations did not mark gender

[8] One sided t-tests performed with equal variance and $p < 0.01$ unless specified otherwise.

[9] A full list of occupations assigned female pronouns can be found in the appendix.

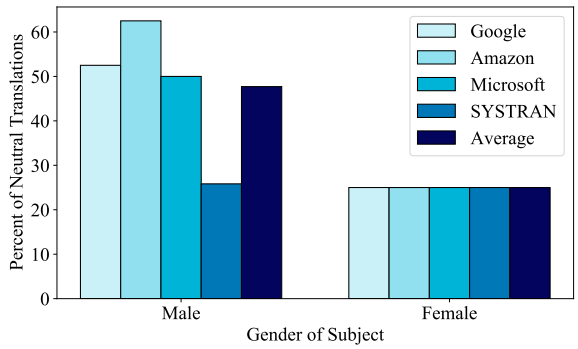


Figure 3: The percentages of translations that used the neutral case according to the gender of the subject per translator as well as the average. While the translations with male subject words had an almost even split, female subject words left gender unmarked only 25% of the time.

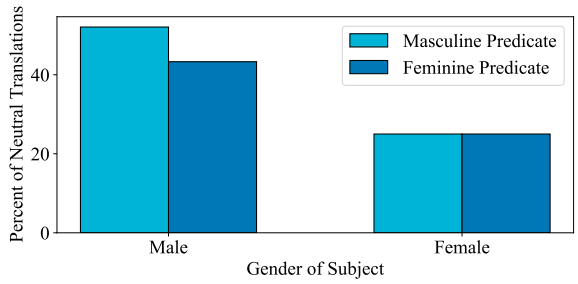


Figure 4: The percent of translations that used the neutral case and didn't preserve gender, across male and female stereotyped predicates, as well as masculine or feminine subjects. For male subject words, gender is significantly more likely to be overtly expressed if the predicate is stereotypically feminine ($p < 0.05$).

and used the neutral form. However, only 25% of the female subject words used the neutral case. This was due to one word, *yeğen* ("niece/nephew"), that remained neutral 100% of the sentences for both male and female subject words. We theorize that this derives from spoken Turkish as *yeğen* ("niece/nephew") is not frequently gender-marked.

Figure 4 demonstrates that when the predicate was stereotypically masculine and the subject word was male, the MT models assumed that the gender of the subject did not need to be overtly expressed, and gender was not preserved 52.1% of the time. For example, "The young men are soccer players" (masculine predicate) did not preserve gender in the translation while "The young men are secretaries" (feminine predicate) did. However, gender was overtly expressed in 56.6% of translations when a stereotypically female predicate was paired with a male subject. Female subject words did not follow this pattern—in fact, for all subject words other than niece/nephew, gender was overtly marked in 75% of the translations. In summary, although male gender was only marked when the content of the sentence deviated from the masculine social norm, female gender was marked in the overwhelming majority of cases, and was consistently treated as aberrational regardless of context.

5 Conclusion

We have examined gender bias exhibited by commercially used MT models in the case of Turkish and English translations. We have shown evidence of overt gender bias through occupation and adjective stereotypes, and covert gender bias through asymmetrical gender and personhood bias. Furthermore, our experiments show consistent evidence of male bias in a neutral context. Male gender was assumed in reference to gender-equal occupations and stereotype-neutral adjectives, and the same phenomenon extends to the manifestation of overt gender markings where male subjects were more likely to be assigned the neutral case. However, when the context was not neutral, stereotype bias routinely affected results across all experiments.

Previous bias mitigation discussions have focused on fair pronoun assignments (Prates et al., 2020; Cho et al., 2019; Baeza-Yates, 2019). Additionally, Google Translate has recently implemented a gender-specific translation feature (Kuczmarski, 2018; Johnson, 2020). While pronoun assignment is a salient and ongoing concern, our study demonstrates how the problem of gender bias can be far more complex. Our experiments show that domain and cultural knowledge are required and these techniques are not necessarily transferable across languages. We advocate for the inclusion of language-specific differences and the design of mitigation models that are linguistically aware and socially grounded. We hope that our work will bring more attention to such interdisciplinary work, prompt continued research in how gender bias is expressed in NLP, and assist with mitigation efforts.

Acknowledgements

We would like to thank Sami Iren for his assistance and insights in verifying our translation data sets for accuracy.

References

Marzieh Babaeianjelodar, Stephen Lorenz, Josh Gordon, Jeanna Matthews, and Evan Freitag. 2020. Quantifying gender bias in different corpora. In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 752–759. Association for Computing Machinery.

Ricardo Baeza-Yates. 2019. Bias on the web. *Symposium on Fairness and Transparency*, page 9.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4356–4364. Curran Associates Inc.

Friederike Braun. 2001. The communication of gender in turkish. In *Gender Across Languages*, pages 283–310. John Benjamins.

Judith Butler. 1990. *Gender trouble: feminism and the subversion of identity*. Routledge, New York, NY.

Won Ik Cho, Ji Won Kim, Seok Min Kim, and Nam Soo Kim. 2019. On measuring gender bias in translation of gender-neutral pronouns. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 173–181. Association for Computational Linguistics.

Marta R. Costa-jussà and Adrià de Jorge. 2020. Fine-tuning neural machine translation on gender-balanced datasets. In *Proceedings of the Second Workshop on Gender Bias in Natural Language Processing*, pages 26–34. Association for Computational Linguistics.

Mostafa Elaraby, Ahmed Y Tawfik, Mahmoud Khaled, Hany Hassan, and Aly Osama. 2018. Gender aware spoken language translation applied to english-arabic. In *2018 2nd International Conference on Natural Language and Speech Processing (ICNLSP)*, pages 1–6. IEEE.

Anna Farkas and Renáta Németh. 2020. How to measure gender bias in machine translation: Optimal translators, multiple reference points. *arXiv preprint arXiv:2011.06445*.

Melvin Johnson. 2020. A scalable approach to reducing gender bias in google translate. *Google AI Blog*.

James Kuczmarski. 2018. Reducing gender bias in google translate. *The Keyword*.

Anne Lauscher and Goran Glavaš. 2019. Are we consistently biased? multidimensional analysis of biases in distributional word vectors. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 85–91. Association for Computational Linguistics.

Ninareh Mehrabi, Thamme Gowda, Fred Morstatter, Nanyun Peng, and Aram Galstyan. 2020. Man is to person as woman is to location: Measuring gender bias in named entity recognition. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, HT '20, page 231–232. Association for Computing Machinery.

Marcelo Prates, Pedro Avelar, and Luís Lamb. 2020. Assessing gender bias in machine translation: a case study with google translate. *Neural Computing and Applications*, 32.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14. Association for Computational Linguistics.

Nuray Sakallı, Beril Türkoğlu, and Abdulkadir Kuzlak. 2018. How are women and men perceived? structure of gender stereotypes in contemporary turkey. *Nesne Psikoloji Dergisi*, 6:309–336.

Danielle Saunders and Bill Byrne. 2020. Reducing gender bias in neural machine translation as a domain adaptation problem. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7724–7736. Association for Computational Linguistics.

Artūrs Stafanovičs, Toms Bergmanis, and Mārcis Pinnis. 2020. Mitigating gender bias in machine translation with target gender annotations. In *Proceedings of the Fifth Conference on Machine Translation*, pages 629–638. Association for Computational Linguistics.

Gabriel Stanovsky, Noah A. Smith, and Luke Zettlemoyer. 2019. Evaluating gender bias in machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1679–1684. Association for Computational Linguistics.

Türkiye Istatistik Kurumu. 2021. İşgücü İstatistikleri, 2020.

U.S. Bureau of Labor. 2020. Labor force statistics from the current population survey.

Celalettin Vatandaş. 2011. Toplumsal cinsiyet ve cinsiyet rollerinin algılanışı. *Istanbul Journal of Sociological Studies*, 0:29 – 56.

Jieyu Zhao, Subhabrata Mukherjee, Saghar Hosseini, Kai-Wei Chang, and Ahmed Hassan Awadallah. 2020. Gender bias in multilingual embeddings and cross-lingual transfer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2896–2907. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. pages 629–634.

A Occupation Data Set Details

This section provides additional details on how the occupation data set was created. The final data set includes matches that are exact matches and matches that are similar. Similar matches fall into one of the following categories:

1. One occupation is a more specific or broad version of it's matching occupation.
2. One occupation uses a slightly different title but describes a similar job.
3. Specifically for educational occupations, the matching occupation describes a different educational level. This helps include occupations that generally exist, but due to different education system setups, are offered at different levels.

Some of the occupation titles have been slightly modified in order to better describe the occupation it matches. These modifications fall into one of the following categories:

1. The occupation title includes punctuation like hyphens or parentheses that describe the occupation. These titles were modified to include the details provided by that occupation.
2. The occupation is split into multiple occupations because it is two separate occupations in the matching country.
3. Specific job details not included in the matching occupation were removed.

Although there were matches, certain occupations were not included for the following reasons:

1. Any religious occupation, due to gender requirements of the majority of those occupations, were not included.
2. Gender specific Turkish occupations. This includes occupations that are either culturally gendered or lexically have gender.
3. Due to different governmental regulations and requirements surrounding gender, military occupations were not included.

Lastly, we list all occupation group names and their abbreviations in Tables 2 and 3.

Abbreviation	SOC Major Group Title
Man.	Management
Bus.	Business and Financial Operations
Comp.	Computer and Mathematical
Arch.	Architecture and Engineering
Eng.	Life and Physical Engineering
Soc.	Community and Social Service
Leg.	Legal
Edu.	Education Training and Library
Art.	Arts, Design, Entertainment, Sports and Media
Hea.	Healthcare Practitioners and Technical
Hea. Sup.	Health Practitioner Support Technologists and Technicians
Ser.	Service
Food	Food Preparation
Bui.	Building and Grounds Cleaning and Management
Per.	Personal Care and Service
Sal.	Sales and Office
Off.	Office Administration Support
Far.	Farming, Fishing and Forestry
Trans.	Transportation and Material Moving
Cons.	Construction and Extraction
Main.	Installation, Maintenance, and Repair

Table 2: Full US SOC occupation titles.

Abbreviation	ISCO Major Group Title
Technicians	Technicians and Associate Professionals
Clerical	Clerical Support Workers
Service	Service and Sales Workers
Agricultural	Skilled Agricultural, Forestry, and Fishery Workers
Trades	Craft and Related Workers
Machine Operators	Plant Machine Operators and Assemblers
Elementary	Elementary Operators

Table 3: Turkish ISCO group names.

B Occupations with Female Generated Pronouns

Table 4 lists all occupations that were assigned female pronouns by at least 3 out of 4 translators.

Occupational Health Spec.	Skin Care Instructor
Barbering Instructor	Emergency Room RN
Registered Nurse	Housekeeping Aide
Surgical Nurse Practitioner	Interior Design Professor
Fashion Designer	CCU Nurse
Certified Diabetes Educator	Bridal Gown Fitter
Cosmetology Instructor	Clinical Nurse Specialist
Makeup Artist	Beautician
Pediatric Registered Nurse	

Table 4: Occupations assigned mostly female pronouns.

The matching Turkish occupation titles can be found in the GitHub[1].

C Sentence Templates in Turkish

Table 5 lists original sentence templates in Turkish.

Original Turkish Template	**English Translation**
O bir <occupation>	He/she is a <occupation>
O bir <adjective>	He/she is <adjective>
O bir <adjective> <occupation>	He/she is a <adjective> <occupation>
O <adjective> birisidir	He/she is someone who is <adjective>

Table 5: Turkish sentence templates. In the third template, the adjective was one of: "çok iyi" (*very good*), "iyi" (*good*), "kötü" (*bad*), or "çok kötü" (*very bad*).

WeaSuL$^{\pi}$: Weakly Supervised Dialogue Policy Learning: Reward Estimation for Multi-turn Dialogue

Anant Khandelwal
India Machine Learning
Amazon
anantkha@amazon.com

Abstract

An intelligent dialogue system in a multi-turn setting should not only generate the responses which are of good quality, but it should also generate the responses which can lead to long-term success of the dialogue. Although, the current approaches improved the response quality, but they over-look the training signals present in the dialogue data. We can leverage these signals to generate the weakly supervised training data for learning dialog policy and reward estimator, and make the policy take actions (generates responses) which can foresee the future direction for a successful (rewarding) conversation. We simulate the dialogue between an agent and a user (modelled similar to an agent with supervised learning objective) to interact with each other. The agent uses dynamic blocking to generate ranked diverse responses and exploration-exploitation to select among the Top-K responses. Each simulated state-action pair is evaluated (works as a weak annotation) with three quality modules: Semantic Relevant, Semantic Coherence and Consistent Flow. Empirical studies with two benchmarks indicate that our model can significantly out-perform the response quality and lead to a successful conversation on both automatic evaluation and human judgement.

1 Introduction

Dialog policy for multi-turn dialogue decides the next best action to take on the environment so as to complete the conversation based on various success criteria. Reinforcement learning can help to learn such a policy where the environment can be users (human or model) and the policy takes action on the environment from which it gets a reward signal (Fatemi et al., 2016; Peng et al., 2017; Chen et al., 2017; Yarats and Lewis, 2018; Lei et al., 2018; He et al., 2018; Su et al., 2018).

Learning a dialogue policy using reinforcement learning can be challenging with humans users, since it requires a large set of samples with a reward to train. Since there are a lot of previous works on neural response generation (Gu et al., 2020; Zhao et al., 2020; Zhang et al., 2019; Xing et al., 2018; Serban et al., 2016) we can model the users also, using any of these encoder-decoder architectures. This helps to simulate the conversations between the simulated user and the agent (policy model) replying to each other (Zhao and Eskenazi, 2016; Dhingra et al., 2016; Shah et al., 2018). Reward signal for policy learning can be as simple as the small constant negative reward at each turn and a large reward at the end (if the goal completes) to encourage shorter conversations (Takanobu et al., 2019).

However, reward estimation for dialogue is challenging, the small constant negative reward at each turn may lead to ending the conversation prematurely. Instead of handcrafting the reward at the end based on success or failure, it is more useful if we can evaluate reward at every turn to guide the policy to dynamically change actions as per the need for the user and end the conversation naturally. With the growing complexity of the system across different topics, it is required to build a more sophisticated reward function to avoid manual intervention for accounting different factors towards conversation success.

In this work, we proposed a novel model for contextual response generation in multi-turn dialogue. The model includes the turn-level reward estimator, which combines the weak supervision signals obtained from three basic modules 1) Semantic Coherence, 2) Consistent Flow, 3) Semantic Relevance. These modules are learned jointly with the response generation model with the counterfactual examples obtained from negative sampling. Leveraging the weak supervision signals obtained from these models, we further update the reward estimator and dialog policy jointly in an alternative way, thus improving each other.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 64–75,
Aberdeen, Scotland, UK, 20-24 September 2021.

Our proposed approach integrates semantic understanding of utterances using encoder-decoder systems with the power of Reinforcement Learning (RL) to optimize long-term success. We test the proposed approach with two benchmarks: DailyDialog (Li et al., 2017b) and PersonaChat (Zhang et al., 2018). Experimental results demonstrate on both datasets indicate that our model can significantly outperform state-of-the-art generation models in terms of both automatic evaluation and human judgment.

2 Related Work

Open-domain dialogue in a multi-turn setting has been widely explored with different encoder-decoder architectures (Gu et al., 2020; Feng et al., 2021; Kottur et al., 2017; Li et al., 2016; Shah et al., 2018; Shang et al., 2015; Vinyals and Le, 2015; Wu et al., 2019; Zhao et al., 2020; Zhong et al., 2019). The basic encoder-decoder architectures like Seq-to-Seq models have been widely extended and modified to generate the generic responses, context modelling and grounding by persona/emotion/knowledge (Li et al., 2015; Xing et al., 2017; Serban et al., 2016; Xing et al., 2018; Zhang et al., 2019, 2018; Zhou et al., 2018; Dinan et al., 2018).

The dialogue literature widely applies reinforcement learning, including the recent ones based on deep architectures (Takanobu et al., 2019, 2020; Li et al., 2020; Takanobu et al., 2020; Li et al., 2020; Gordon-Hall et al., 2020a,b). But these task-oriented RL dialogue systems often model the dialogue with limited parameters and assumptions specific to the dataset, targeted for that task. The dataset includes hand-built templates with state, action and reward signals designed by humans for each new domain making this setting difficult for extending these to open domain dialogue systems.

Our goal in this work is to integrate the state-of-the-art encoder-decoder architectures like in Gu et al. (2020); Zhao et al. (2020); Csaky and Recski (2020) and reinforcement learning paradigms to efficiently learn the dialogue policy optimized for long-term success in the multi-turn dialogue scenarios. We are recently inspired by the works in Takanobu et al. (2019); Li et al. (2020, 2016) to jointly learn the reward function and dialogue policy, and reduce the effort and cost for manual labelling the conversations for building the reward model. Specifically, we leverage the weak supervision inspired from Chang et al. (2021a,b) to generate the labelled dataset to facilitate this joint learning and building reward estimation model.

3 Approach

We represent dialog sessions $\mathcal{D} = \{\tau_1, \tau_2, \tau_3, ...\tau_n\}$ where each dialog session τ represents the trajectory of state-action pairs as $\{s_0^u, a_0^u, s_0, a_0, s_1^u, a_1^u, s_1, a_1, ...\}$. The user in our case is a simulator which utters a response a^u given the state s^u denoted as $\mu(a^u, e^u|s^u)$ where e^u denotes the binary signal indicating the end of a dialog session, in that case the response a^u is empty. The dialog policy $\pi_\theta(a|s)$ decides the action a according to the current state s after the agent interacts with the user simulator μ. At each time, the state given to the either dialog party is updated after recording the action uttered by the other party. The reward estimator f evaluates the quality of response/action uttered by the dialog policy π. The dialog policy π is based on the BERT (Devlin et al., 2019) encoder-decoder model and the reward function f is the MLP model parameterized by θ and ω respectively. We have modeled the user simulator exactly in the same way as the agent but trained only using supervised learning objective.

In the subsequent section, we will introduce the components action, state, policy, quality modules and reward estimator. Further, sections explain the setup we have used for weakly supervised learning and, finally, the experimental results.

3.1 Action

An action a is the dialogue utterance generated by the encoder-decoder model as shown in Figure 1. The model takes as input the context history (state), and outputs the probability distribution over a set of possible actions denoted as $\pi_\theta(a|s)$ parameterized by θ. The user simulator generates the action a^u, policy generates the action a, and the input state for the agent and the user is s and s^u respectively.

3.2 State

The state is the past conversation history between an agent and a user denoted as, $s_t = \{q_1, a_1, q_2, a_2, q_3, a_3, ..., q_t\}$. The state for an agent and a user are differently denoted as s and s^u respectively. Let's say the agent utterances are denoted by a's, then state, $s = s_t$ and the agent utters a_t. Similarly, the user state

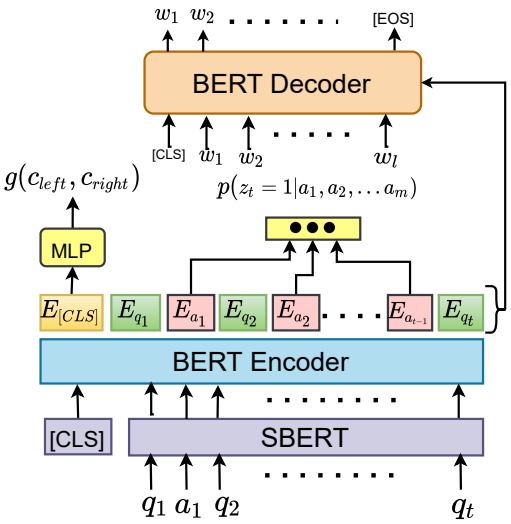


Figure 1: BERT based Encoder-Decoder with Semantic Coherence and Relevance. Similarly, Consistent Flow loss is also calculated using encoder.

$s_t^u = \{q_1, a_1, q_2, a_2, q_3, a_3, ..., q_t, a_t\}$ and the user utters q_{t+1}. Each of the utterances is mapped to a fixed-length sentence vector using SBERT (Reimers and Gurevych, 2019).

3.3 Dialogue Policy

The dialogue policy takes the form of a BERT based encoder-decoder (i.e. $\pi_\theta(a|s)$) (Gu et al., 2020) as shown in Figure 1. Similar to Xu et al. (2020), we have used the BERT based encoder and transformer decoder, but instead of feeding the utterance at word level, we instead fed the utterance representation (obtained from SBERT) into the encoder. The encoder takes as input the previous context history as s_t and output the response a_t at the output of the decoder.

3.4 User Simulator

We have modelled the user simulator in exactly the same way as the BERT based encoder-decoder shown in Figure 1. However, the user simulator is trained only (with supervised learning objective) for utterances in dialog corpus and predicting user response (Gu et al., 2020).

3.5 Conversation Quality Modules

We calculate the reward for each state-action pair (see Section. 3.8) and use this signal to train the dialogue policy so that it can avoid reaching bad states so as to reach the successful end of the conversation between a user and an agent. We have leveraged the signals from three basic modules, namely, Semantic Coherence, Consistent Flow and Semantic Relevance (which are jointly learned with the dialogue policy). For each of the three modules, the data for the positive class is obtained from the source corpus while for the negative class it has been generated dynamically during training. We describe each of the three modules in the following sections.

3.5.1 Semantic Relevance

We need to filter out the utterances generated with high confidence by the dialog policy but are semantically irrelevant to the previous context. To quantify such a characteristic, we modeled the general response relevance prediction task which utilizes the sequential relationship of the dialog data fed to the encoder side of BERT encoder-decoder framework. Since, the task of semantic relevance is to match the two sequences of conversation, so instead of matching the context and response, we have measured the relevance of two fragments of dialogue session.

Specifically, given a context $c = \{q_1, a_1, q_2, a_2, ...q_m\}$, we randomly split c into two consecutive pieces $c_{left} = \{q_1, a_1, q_2, a_2, ...q_t, a_t\}$ and $c_{right} = \{q_{t+1}, a_{t+1}, ...q_m\}$. Similar to Xu et al. (2020), we replaced the left or right part with the sampled piece from the corpus. Also, we additionally generate the negative samples by internal shuffling in the left or right part. The whole model is trained like a classifier with corresponding labels $y_{sr} \in \{0, 1\}$. Since the individual utterances are fed after obtaining their vector representation, the aggregated representation of two pieces is represented by E^{sr}_{CLS} over which the non-linear transformation is applied, the score for semantic relevance is given by $g(c_{\text{left}}, c_{\text{right}})$, and similar to Xu et al. (2020), it has been trained using the binary cross-entropy loss as:

$$L_{sr} = -y_{sr} \log(g(c_{\text{left}}, c_{\text{right}})) - (1 - y_{sr}) \log(1 - g(c_{\text{left}}, c_{\text{right}})) \quad (1)$$

3.5.2 Semantic Coherence

The response generated should be rewarded only if it is coherent despite having adequate content. This makes the model to generate the coherent responses while avoiding the incoherent ones. Specifically, given a context $c = \{q_1, a_1, q_2, a_2, ...q_m\}$, we randomly select any of the agent response at time t, denoted as a_t, and replace it with any random

utterance from the corpus. We also generate the incoherent samples by internal shuffling of bi-grams. The incoherent utterance is labelled as $y_t^{coh} = 0$ and coherent samples as $y_t^{coh} = 1$. The semantic coherence model is also trained like a classifier for each of the utterance representations obtained at the output of BERT encoder as shown in Figure 1. The probability of the t-th utterance being incoherent is given as:

$$p(z_t = 1|a_1, .., a_t) = softmax(W_{coh}E_{a_t}+b_{coh}) = \frac{exp(W_{coh}E_{a_t} + b_{coh})}{\sum_{l=1}^{m} exp(W_{coh}E_{a_l} + b_{coh})} \quad (2)$$

and the loss function is given as:

$$L_{coh} = -\sum_{t=1}^{m} z_t \log p(z_t = 1|a_1, a_2.....a_m) \quad (3)$$

3.5.3 Consistent Flow

We want the agent to continuously add the information to keep the conversation going in the forward direction. To determine the flowing conversation, we take the cosine similarity between the last two agent utterances denoted as $E_{a_{i-1}}$ and E_{a_i} denoted as $g(a_{i-1}, a_i)$, and we measure the similarity with randomly sampled utterance v in place of a_{i-1} given as $g(a_{i-1}, v)$. We would like $g(a_{i-1}, a_i)$ to be larger than $g(a_{i-1}, v)$ by at least a margin Δ and define the learning objective as a hing loss function:

$$L_{cf} = max\{0, \Delta - g(a_{i-1}, a_i) + g(a_{i-1}, v)\} \quad (4)$$

3.6 Joint Training of Agent and Reward Modules

To initialize the parameters of agent and reward modules $\mathcal{M}$ ={Semantic Relevance, Semantic Coherence, Consistent Flow}, we used the supervised learning objective since all the state-action pairs obtained from the pre-training corpus are the ground-truth and can be used as close approximation for further fine-tuning on other dialog corpus. We used the pre-training corpus $\mathcal{P}$ as Gutenberg dialog corpus (Csaky and Recski, 2020). Since the agent model in our case is based on BERT encoder-decoder parameterized by θ similar to Gu et al. (2020), the probability of generating agent's response **a** is given as:

$$p_\theta(\mathbf{a}|\mathbf{s}) = \prod_{j=1}^{N} p_\theta(a_j|a_{<j}, \mathbf{s}), \quad (5)$$

where a_j is the j-th word generated at the output of decoder and **s** is the whole context history utterances fed to the encoder and N is the maximum sequence length of decoder. The loss function for generating agent response **a** is given as:

$$L_a = J(\theta) = -\sum_{i=1}^{N} \log p_\theta(a_j|a_{<j}, \mathbf{s}) \quad (6)$$

The joint loss function is defined as:

$$L_{full} = L_a + \alpha * (L_{sr} + L_{coh} + L_{cf}) \quad (7)$$

The policy π_θ is also parameterized by θ, and the probability of action a is given by $\pi_\theta(a|s)$ similar to $p_\theta(a|s)$, since the probability distribution is learned only from (s, a) pairs obtained from the corpus with human demonstrations. It is a good approximation to initialize the parameters of policy $\pi_\theta(a|s)$ with parameters of $p_\theta(a|s)$. Furthermore, we update the policy π_θ (Step 13 in the Algorithm. 1) to avoid actions a which do not lead to rewarding conversations.

3.7 Dialogue Simulation between Agent and User

We setup simulation between virtual agent and user, and let them take turns talking to each other. The simulation is started with a starter utterance obtained from the dialog samples D_H (Step 5 of Algorithm 1) and fed to the agent, it then encodes the utterance and generates the response a, the state s^u is then updated with previous history and fed to the user model to obtain the next response a^u. The response a^u is appended to s^u to obtain the updated state s. Similarly, the process is repeated until one of the following conditions occurs after a few number of turns[1]: a) When agent starts to produce dull responses like "I don't know" [2]. b) When agent starts to generate repetitive response consecutively [3] c) Or, the conversation achieved the maximum number of turns handled by agent and user models.[4]

[1]The number of turns after these rules applied is average number of turns in the corpus

[2]Used simple rule matching method with 9 phrases collected from the corpus, instead of having false positives and negatives this works well in practice.

[3]If by rule two consecutive utterances matched more than 80% it is said to be repetitive.

[4]The maximum number of turn is set as 20.

3.8 Weakly Supervised Learning Algorithm

Learning with weak supervision is widely used with the rise of data-driven neural approaches (Ratner et al., 2020; Mrkšić et al., 2017; Chang et al., 2020; Bach et al., 2017; Wu et al., 2018; Chang et al., 2021a). Our approach incorporates a similar line of work by providing noisy text to a pre-trained model which incorporates prior knowledge from general-domain text and small in-domain text (Peng et al., 2020; Chen et al., 2019; Harkous et al., 2020) and use it as a weak annotator similar to Ratner et al. (2020). The primary challenge with the synthetic data is the noise introduced during the generation process, and the noisy labels tend to bring little to no improvement (Frénay and Verleysen, 2013). To train on such noisy data, we employ three step training process: a) pre-training b) generate data with weighted categories c) fine-tuning similar to Chang et al. (2021a); Dehghani et al. (2017).

Step 1: Pre-train Generation and Quality Modules Jointly. This step involves pre-training the agent with quality modules jointly as explained in Section 3.6. Quality modules trained on clean data as well as automatically generated negative samples by random sampling. These modules are further fine-tuned on the sampled dialogues from target dialogue corpus at each training iteration. Similarly, we initialized the user also by supervised training on the pre-training dialogue corpus with fine-tuning on target dialogue corpus. (see steps 2-7 of Algorithm 1). The fine-tuning steps make use of continual learning to avoid catastrophic forgetting (Madotto et al., 2020; Lee, 2017).

Step 2: Generates the Weakly Labelled data with Reward categories. After the models are initialized with trained parameters, the dialogue simulation has been started between the agent and the user (see Section. 3.7) to interact with each other and generates the synthetic data with annotated scores with each quality module for every state-action pair in sampled dialogues. During dialogue simulation, we employ Dynamic Blocking mechanism(Niu et al., 2020) to generate novel words and paraphrased responses. Specifically, we generate Top-7 response at each turn and set the agent to exploration for 60 percent of the times and for the rest of the times it exploits by selecting the response from top two ranked responses. We specifically filter the state-action pairs into three reward categories namely, *VeryHigh*, *High* and *Low*. For the state-action pairs whose scores by each module are greater than or equal to 0.8 are put into the *VeryHigh* category. Other, state-action pairs whose scores by each module are between 0.6 and 0.8 are put into the *High* reward category. The rest of all state-action pairs are put into the *Low* reward category. Additionally, we include state-action pairs sampled from target dialog corpus in Step 1. into the *VeryHigh* category.

Step 3: Update the reward estimator and policy. The reward estimator maximizes the log likelihood state-action pairs of higher rewards than the lower ones. The reward estimator f_ω, parameterized by ω, and let's say H, V and L represents the collection of all state action pairs of *High*, *VeryHigh* and *Low* reward category respectively.

$$\begin{aligned}
\omega^* &= \arg\max \mathbb{E}_{(s_k,a_k)\sim\{H,V,L\}}[f_\omega(s_k,a_k)] \\
f_\omega(s_k,a_k) &= \log p_\omega(s_k,a_k) = \log \frac{e^{R_\omega(s_k,a_k)}}{Z_\omega} \\
R_\omega(s_k,a_k) &= \sum_{t=k}^{T} \gamma^{t-k} r_\omega(s_t,a_t) \\
Z_\omega &= \sum_{\forall(s_k,a_k)} e^{R_\omega(s_k,a_k)}
\end{aligned} \tag{8}$$

where f models state-action pairs of H, V and L category as a Boltzmann distribution (Takanobu et al., 2019). The cost function for reward estimator in terms of trajectories obtained from respective reward categories is given as:

$$\begin{aligned}
J_f(\omega) = &-0.5 * KL(p_H(s,a) \parallel p_\omega(s,a)) \\
&- KL(p_V(s,a) \parallel p_\omega(s,a)) \\
&+ KL(p_L(s,a) \parallel p_\omega(s,a))
\end{aligned} \tag{9}$$

It minimize the KL-divergence between reward distribution and the state-action pairs of high and very high reward but maximize the distribution from the ones with low category. The gradient yields:

$$\begin{aligned}
\bigtriangledown_\omega J_f = 0.5 * \mathbb{E}_{(s,a)\sim H}[\bigtriangledown_\omega f_\omega(s,a)] \\
+\mathbb{E}_{(s,a)\sim V}[\bigtriangledown_\omega f_\omega(s,a)] - \mathbb{E}_{(s,a)\sim L}[\bigtriangledown_\omega f_\omega(s,a)]
\end{aligned} \tag{10}$$

Since, the dialog policy is required to put the actions atleast to that of high category, i.e. maximize the entropy regularized expected reward ($\mathbb{E}_\pi[R] + H(\pi)$) which is effectively minimizes

the KL divergence between the policy distribution and Boltzmann distribution.

$$J_\pi(\theta) = -KL(\pi_\theta(a|s) \parallel p_\omega(s,a)) \\ = \mathbb{E}_{(s,a)\sim\pi}[f_\omega(s,a) - \log \pi_\theta(a|s)] \\ = \mathbb{E}_{(s,a)\sim\pi}[R_\omega(s,a)] - \log Z_\omega + H(\pi_\theta) \quad (11)$$

where the term $\log Z_\omega$ is independent to θ, and $H(\cdot)$ denotes the entropy of a model. Using likelihood ratio trick the gradient for policy is given as:

$$\nabla_\theta J_\pi = \mathbb{E}_{(s,a)\sim\pi}[(f_\omega(s,a) \\ - \log \pi_\theta(a|s))\nabla_\theta \log \pi_\theta(a|s)]. \quad (12)$$

Hence, the reward is $r_\omega(s,a) = f_\omega(s,a) - \log \pi_\theta(a|s)$ for each state-action pair and the loss function re-written as:

$$J_\pi(\theta) = \mathbb{E}_{(s,a)\sim\pi}[\sum_{k=t}^{T} \gamma^{k-t}(f_\omega(s_k,a_k) \\ - \log \pi_\theta(a_k|s_k))] \quad (13)$$

Like in Takanobu et al. (2019) the reward estimator f_ω includes the shaping term. Formally, we include next state s_{t+1} also instead of just (s_t, a_t)

$$f_\omega(s_t, a_t, s_{t+1}) = g_\omega(s_t, a_t) + \gamma h(s_{t+1}) - h(s_t) \quad (14)$$

where h is the MLP network with input as pre-sigmoid scores from each quality modules, and g_ω is also the MLP network with input as the concatenation of E_{CLS} as state vector and SBERT sentence embedding of action a.

Algorithm 1 Dialogue Policy Learning

Require: Pre-Training corpus P, Dialogue Corpus $\mathcal{D}$.

1: Modules $\mathcal{M}$ = {Semantic Relevance, Semantic Coherence, Consistent Flow}
2: Do Agent training on $\mathcal{P}$ as in Section 3.6 jointly with modules $\mathcal{M}$
3: User μ supervised training on $\mathcal{P}$.
4: **for each** *training iteration* **do**
5: Sample dialogues $\mathcal{D}_H$ from $\mathcal{D}$ randomly.
6: Fine-tune user simulator μ on $\mathcal{D}_H$.
7: Fine-tune Agent and $\mathcal{M}$ on $\mathcal{D}_H$ jointly.
8: Collect dialog samples $\mathcal{D}_\pi$ by executing the dialog policy π and interacting with μ, $a^u \sim \mu(\cdot|s^u)$, $a \sim \pi(\cdot|s)$ where s and s^u is updated each time after getting response from user and agent respectively.
9: Get weak annotation scores for all $(s,a) \in \mathcal{D}_\pi$ from each of the modules $\mathcal{M}$.
10: Filtering the (s,a) pairs into {*VeryHigh*, *High* and *Low*} reward categories.
11: Update the reward estimator f by minimizing J_f w.r.t ω (Eq.10)
12: Compute reward for each $(s,a) \in \mathcal{D}_\pi$ as,

$$\hat{r} = f_\omega(s_t, a_t, s_{t+1}) - \log \pi(a_t|s_t)$$

13: Update the policy π_θ by minimizing J_π w.r.t θ (Eq. 13).
14: **end for**

4 Experiments

We conduct experiments on DailyDialog (Li et al., 2017b), PersonaChat (Zhang et al., 2018) and used Gutenberg Dialogue Dataset (Csaky and Recski, 2020) as a pre-training corpus. We compare our model performance with baselines on various aspects of response quality.

4.1 Datasets

We considered DailyDialog (Li et al., 2017b) and PersonaChat (Zhang et al., 2018) which are open domain dialog corpus to evaluate our system. DailyDialog contains conversation revolving around various topics pertaining to daily life, and PersonaChat contains conversations between people with their respective persona profiles. These dialogues can be of varying length, we limit the maximum length to 20, that can be fed to the BERT Encoder-Decoder model. Since average length of DailyDialog is 7.9 and that of PersonaChat is 9.4, so most of the dialogues fit easily without truncation from the history. For rest of the dialogues, it can be slided across to include the more recent utterances and remove it from the starting. Since we are mapping the utterances to their corresponding vectors using SBERT, the length of individual utterances truncated automatically and retain only first 512 word pieces in case of longer utterances. For pre-training corpus the vocabulary is limited to 100,000 while the vocabularies for DailyDialog and PersonaChat are 25,000 and 32,768 respectively.

4.2 Baselines

We select various multi-turn response generation baselines. The baselines which are not included pre-training are (1) **HRED** : Hierarchical encoder-decoder framework (Serban et al., 2016)

(2) **VHRED** : an extension of HRED that generates response with latent variables (Serban et al., 2017) (3) **HRAN** : Hierarchical attention mechanism based encoder-decoder framework (Xing et al., 2018) (4) **ReCoSa** : Hierarchical transformer based model (Zhang et al., 2019) (5) **SSN**: dialogue generation learning with self-supervision signals extracted from utterance order (Wu et al., 2019) (6) **Transformer-Auxiliary Tasks**: A recent state-of-the are model leaning language generation with joint learning of transformer with auxiliary tasks (Zhao et al., 2020). The another two baselines from Csaky and Recski (2020) which involve pre-training on the Gutenberg corpus are: (1)**Transformer** : 50M parameters version and (2) **GPT-2** : Pre-trained model with version of 117M parameters. The repository[5] contains these two trained models.

4.3 Evaluation Metrics

We evaluate the performance of our model on various aspects of response quality using both automatic and human evaluation. Although, most of the automatic metrics poorly correlate with human evaluation (Liu et al., 2016), and the recently proposed metrics (Li et al., 2017a; Lowe et al., 2017; Tao et al., 2018) are harder to evaluate than perplexity and BLEU (Papineni et al., 2002). Additionally, human evaluation has its inherent limitation of bias, cost and replication difficulty (Tao et al., 2018). Due to this consensus, some used only automatic metrics (Xing and Fernández, 2018; Xu et al., 2018b) and some used only human evaluation (Krause et al., 2017; Fang et al., 2018) while some used both (Shen et al., 2018; Xu et al., 2018a; Baheti et al., 2018; Ram et al., 2018).

We mainly used the automatic metrics using the DIALOG-EVAL repository[6], it contains 17 different metrics, but we measure only a few metrics to facilitate the comparison with the published baselines results. We specifically follow (Zhao et al., 2020) to measure automatic evaluation and human evaluation. For response content quality we measured BLEU-4 (Papineni et al., 2002) and Perplexity(PPL) (Sutskever et al., 2014). Like in Zhao et al. (2020) used embedding metrics average (AVG), extrema (EXT), and greedy (GRE) measuring similarity between response and target embedding. Similar to Zhao et al. (2020) we also measured the informativeness of responses with distinct-1 and distinct-2 that are calculated as the ratios of distinct unigrams and bigrams.

Since our main objective is not to judge the response quality but to predict the response for long-term success of dialogue. We follow the guidelines as in Li et al. (2016) to explore both single-turn and multi-turn settings. We picked 500 dialogues from the test set and asked 3 native speakers for their judgement. In the first setting, we asked judges to pick the better response among the one generated by our model and a baseline model (**Pre-Trained GPT2**) based on various criteria like answerability and semantics. In the second setting, in case of multi-turn we used 200 simulated conversations between RL agent and a user model to judge the whole conversation for responses uttered by agent. In a complete end-to-end conversation we asked the judges to decide which of the simulated conversations are of higher quality. To compare against the RL model we employ baseline model to simulate the 200 conversations with the same starter utterance used by RL model. Automatic and Human evaluation are shown in Table. 1 and 2 respectively.

4.4 Results and Discussions

Table. 1 reports automatic evaluation metrics on the baseline and the proposed model. Our model outperforms for most of the metrics on both datasets. Since our main idea is to generate the responses for successful conversation in the long run than just evaluating the response quality at each of the turn. This is the main reason of why our model outperforms on both distinct-1 and distinct-2 metrics, in comparison to Transformer-auxiliary task model which also trained jointly with the similar tasks but lacks fine-tuning with the weak supervision signals indicate that an additional training with weakly labelled data improves the generalization performance. Although, we see the perplexity also improves since our model is generating the responses more like humans to optimize the conversation in long run. Similarly, embedding metrics also shown the improvement but little on average since it capturing the sense but due to length mismatch which occurs owing to the fact that our model is generating more novel words with futuristic sense. However, Distinct-{1,2} scores shows improvement because of the large pre-trained vocabulary, it gives the model more flexibility to generate novel words without disturbing the sense of the sentence.

[5]https://github.com/ricsinaruto/gutenberg-dialog
[6]https://github.com/ricsinaruto/ dialog-eval

Dataset	Model	PPL	BLEU	Distinct-1	Distinct-2	Average	Greedy	Extrema
DailyDialog	HRED	56.22	0.535	1.553	3.569	81.393	65.546	48.109
	HRAN	47.23	0.447	1.953	7.400	83.460	67.239	49.599
	VHRED	44.79	0.997	1.299	6.113	83.866	67.186	48.570
	SSN	44.28	1.250	2.309	7.266	72.796	73.069	44.260
	ReCoSa	42.34	1.121	1.987	10.180	84.763	67.557	48.957
	Transformer-Auxiliary Tasks	38.60	1.658	3.457	14.954	85.224	69.518	49.069
	Pre-Trained Transformer	-	11.5	2.92	14.7	55.1	53.5	59.8
	Pre-Trained GPT2	-	12.8	4.07	25.9	56.8	54.0	59.6
	Our Model	**20.13**	**15.171**	**6.316**	**28.422**	85.417	**73.118**	**61.539**
	Our Model w/o weak supervision	20.51	14.718	4.611	26.752	**86.481**	73.003	59.911
PersonaChat	HRED	46.04	1.279	0.164	0.450	83.329	65.546	48.109
	HRAN	41.94	1.997	0.235	0.771	82.850	67.239	49.599
	VHRED	42.07	2.181	0.312	1.915	82.995	67.186	48.570
	SSN	47.90	2.288	0.637	2.623	85.002	73.069	44.260
	ReCoSa	34.19	2.258	0.915	4.217	83.963	67.557	48.957
	Transformer-Auxiliary Tasks	33.23	2.434	1.279	5.816	83.632	69.518	49.069
	Pre-Trained Transformer	-	15.5	1.04	4.8	51.3	57.5	57.1
	Pre-Trained GPT2	-	15.3	1.82	12.9	53.6	55.9	55.8
	Our Model	**19.78**	**16.651**	**2.434**	**13.912**	84.941	**73.081**	**59.241**
	Our Model w/o weak supervision	21.49	16.017	2.318	13.274	**85.018**	72.438	58.816

Table 1: Automatic metrics comparison with baselines. Results in bold indicate the best performing model on the corresponding metrics.

DailyDialog			
Setting	**RL-Win**	**RL-Lose**	**Tie**
Single-Turn general quality	0.41	0.28	0.31
Single-Turn ease to answer	0.55	0.12	0.33
Multi-turn general quality	0.76	0.13	0.11
PersonaChat			
Setting	**RL-Win**	**RL-Lose**	**Tie**
Single turn general quality	0.36	0.22	0.42
Single-Turn ease to answer	0.51	0.14	0.35
Multi-turn general quality	0.71	0.17	0.12

Table 2: Human Evaluation Results. Ratios are calculated after taking majority vote among the decisions made by three judges.

We also note the results for our model without weak supervision training, namely, **Our Model w/o Weak Supervision**, this model just fine-tunes on the DailyDialog (Li et al., 2017b) and PersonaChat (Zhang et al., 2018) without generating the weak labelled data. Clearly, the distinct-1 and distinct-2 metrics are lower than the proposed model, because the model tends to generate the repetitive words more frequently. Similarly, the embedding metrics and PPL does not show any improvement over the proposed model except on embedding metric based on Average. However, it performs well on BLEU scores since it learns well to reproduce the responses as in the ground truth but not optimized for a successful conversation in the long run.

Table 1 also reports the results of another two baselines which are pre-trained models on Gutenberg Dialogue Corpus (Csaky and Recski, 2020). These models are fine-tuned on DailyDialog and PersonaChat dataset respectively. These models although improved much on BLEU scores and distinct-1 and distinct-2 scores since it gets the larger vocab and more enhanced training for learning the language structure. But lags in the embedding metrics indicating the response quality is low.

Table 2 reports the human evaluation results, the objective for which our model training is to generate the response for a successful conversation in the long run for the multi-turn scenario. Clearly, the evaluation results are up to our expectation, since the RL system does not bring a significant boost in single-turn response quality than the case of multi-turn setting.

5 Conclusions

We proposed a weak supervision framework for policy and reward estimation for long-term success of the dialogue by simulating the conversation between a virtual agent and user. Empirical studies

on two benchmarks proves the effectiveness of our approach.

References

Stephen H Bach, Bryan He, Alexander Ratner, and Christopher Ré. 2017. Learning the structure of generative models without labeled data. In *International Conference on Machine Learning*, pages 273–282. PMLR.

Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. *arXiv preprint arXiv:1809.01215*.

Ernie Chang, David Ifeoluwa Adelani, Xiaoyu Shen, and Vera Demberg. 2020. Unsupervised pidgin text generation by pivoting english data and self-training. *arXiv preprint arXiv:2003.08272*.

Ernie Chang, Vera Demberg, and Alex Marin. 2021a. Jointly improving language understanding and generation with quality-weighted weak supervision of automatic labeling. *arXiv preprint arXiv:2102.03551*.

Ernie Chang, Xiaoyu Shen, Dawei Zhu, Vera Demberg, and Hui Su. 2021b. Neural data-to-text generation with lm-based text augmentation. *arXiv preprint arXiv:2102.03556*.

Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu. 2017. Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2454–2464.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2019. Few-shot nlg with pre-trained language model. *arXiv preprint arXiv:1904.09521*.

Richard Csaky and Gabor Recski. 2020. The gutenberg dialogue dataset. *arXiv preprint arXiv:2004.12752*.

Mostafa Dehghani, Arash Mehrjou, Stephan Gouws, Jaap Kamps, and Bernhard Schölkopf. 2017. Fidelity-weighted learning. *arXiv preprint arXiv:1711.02799*.

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

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2016. Towards end-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*.

Hao Fang, Hao Cheng, Maarten Sap, Elizabeth Clark, Ari Holtzman, Yejin Choi, Noah A. Smith, and Mari Ostendorf. 2018. Sounding board: A user-centric and content-driven social chatbot. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 96–100, New Orleans, Louisiana. Association for Computational Linguistics.

Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. 2016. Policy networks with two-stage training for dialogue systems. *arXiv preprint arXiv:1606.03152*.

Shaoxiong Feng, Xuancheng Ren, Kan Li, and Xu Sun. 2021. Multi-view feature representation for dialogue generation with bidirectional distillation. *arXiv preprint arXiv:2102.10780*.

Benoît Frénay and Michel Verleysen. 2013. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869.

Gabriel Gordon-Hall, Philip John Gorinski, and Shay B Cohen. 2020a. Learning dialog policies from weak demonstrations. *arXiv preprint arXiv:2004.11054*.

Gabriel Gordon-Hall, Philip John Gorinski, Gerasimos Lampouras, and Ignacio Iacobacci. 2020b. Show us the way: Learning to manage dialog from demonstrations. *arXiv preprint arXiv:2004.08114*.

Xiaodong Gu, Kang Min Yoo, and Jung-Woo Ha. 2020. Dialogbert: Discourse-aware response generation via learning to recover and rank utterances. *arXiv preprint arXiv:2012.01775*.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. *arXiv preprint arXiv:2004.06577*.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. 2018. Decoupling strategy and generation in negotiation dialogues. *arXiv preprint arXiv:1808.09637*.

Satwik Kottur, Xiaoyu Wang, and Vítor Carvalho. 2017. Exploring personalized neural conversational models. In *IJCAI*, pages 3728–3734.

Ben Krause, Marco Damonte, Mihai Dobre, Daniel Duma, Joachim Fainberg, Federico Fancellu, Emmanuel Kahembwe, Jianpeng Cheng, and Bonnie

Webber. 2017. Edina: Building an open domain socialbot with self-dialogues. *arXiv preprint arXiv:1709.09816*.

Sungjin Lee. 2017. Toward continual learning for conversational agents. *arXiv preprint arXiv:1712.09943*.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017a. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017b. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of The 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*.

Ziming Li, Sungjin Lee, Baolin Peng, Jinchao Li, Shahin Shayandeh, and Jianfeng Gao. 2020. Guided dialog policy learning without adversarial learning in the loop. *arXiv preprint arXiv:2004.03267*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Ryan Lowe, Michael Noseworthy, Iulian V Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. *arXiv preprint arXiv:1708.07149*.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, and Zhiguang Wang. 2020. Continual learning in task-oriented dialogue systems. *arXiv preprint arXiv:2012.15504*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, Vancouver, Canada. Association for Computational Linguistics.

Tong Niu, Semih Yavuz, Yingbo Zhou, Huan Wang, Nitish Shirish Keskar, and Caiming Xiong. 2020. Unsupervised paraphrase generation via dynamic blocking. *arXiv preprint arXiv:2010.12885*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. *arXiv preprint arXiv:1704.03084*.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. *arXiv preprint arXiv:2002.12328*.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. 2018. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2020. Snorkel: Rapid training data creation with weak supervision. *The VLDB Journal*, 29(2):709–730.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3295–3301. AAAI Press.

Pararth Shah, Dilek Hakkani-Tur, Bing Liu, and Gokhan Tur. 2018. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 41–51.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Xiaoyu Shen, Hui Su, Wenjie Li, and Dietrich Klakow. 2018. Nexus network: Connecting the preceding and the following in dialogue generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4316–4327.

Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen. 2018. Discriminative deep dyna-q: Robust planning for dialogue policy learning. *arXiv preprint arXiv:1808.09442*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

Ryuichi Takanobu, Runze Liang, and Minlie Huang. 2020. Multi-agent task-oriented dialog policy learning with role-aware reward decomposition. *arXiv preprint arXiv:2004.03809*.

Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. 2019. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. *arXiv preprint arXiv:1908.10719*.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Jiawei Wu, Xin Wang, and William Yang Wang. 2019. Self-supervised dialogue learning. *arXiv preprint arXiv:1907.00448*.

Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2018. Learning matching models with weak supervision for response selection in retrieval-based chatbots. *arXiv preprint arXiv:1805.02333*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Chen Xing, Yu Wu, Wei Wu, Yalou Huang, and Ming Zhou. 2018. Hierarchical recurrent attention network for response generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Yujie Xing and Raquel Fernández. 2018. Automatic evaluation of neural personality-based chatbots. *arXiv preprint arXiv:1810.00472*.

Can Xu, Wei Wu, and Yu Wu. 2018a. Towards explainable and controllable open domain dialogue generation with dialogue acts. *arXiv preprint arXiv:1807.07255*.

Ruijian Xu, Chongyang Tao, Daxin Jiang, Xueliang Zhao, Dongyan Zhao, and Rui Yan. 2020. Learning an effective context-response matching model with self-supervised tasks for retrieval-based dialogues.

Xinnuo Xu, Ondřej Dušek, Ioannis Konstas, and Verena Rieser. 2018b. Better conversations by modeling, filtering, and optimizing for coherence and diversity. *arXiv preprint arXiv:1809.06873*.

Denis Yarats and Mike Lewis. 2018. Hierarchical text generation and planning for strategic dialogue. In *International Conference on Machine Learning*, pages 5591–5599. PMLR.

Hainan Zhang, Yanyan Lan, Liang Pang, Jiafeng Guo, and Xueqi Cheng. 2019. Recosa: Detecting the relevant contexts with self-attention for multi-turn dialogue generation. *arXiv preprint arXiv:1907.05339*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*.

Yufan Zhao, Can Xu, and Wei Wu. 2020. Learning a simple and effective model for multi-turn response generation with auxiliary tasks. *arXiv preprint arXiv:2004.01972*.

Peixiang Zhong, Di Wang, and Chunyan Miao. 2019. An affect-rich neural conversational model with biased attention and weighted cross-entropy loss. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7492–7500.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

A Implementation Details

Our implementation uses the open source Huggingface Transformer repository (Wolf et al., 2020). Specifically, we have used the base version from sentence transformers pre-trained on millions of paraphrase examples, named as '*paraphrase-distilroberta-base-v1*'. The encoder-decoder framework is initialized with the base version '*bert-base-uncased*'but with configuration of smaller size. The smaller sized model reduces the '*bert-base-uncased*'configuration to 6 transformer layers, has a hidden size of 768, and contains 2 attention heads, {L=6, H=768, A=2}. Similar to Gu et al. (2020) we sum the position embeddings to the output sentence embeddings of size 768 to indicate the user or agent utterances. Odd ones indicate the user utterances and even ones are that of an agent. The MLP network for semantic relevance and semantic coherence used a hidden dimension of 128. The Δ has been set to best value of 0.54 after performing a grid search in the range of $\{0.4, 0.7\}$ with step size of 0.02. The reward estimator models g_ω using two hidden layers of size 512 and 256 respectively. And, h is modelled using a single hidden layer of size one. In each training iteration the policy and reward estimator are updated with continual learning to avoid catastrophic forgetting mechanism using EWC modified loss, the λ value used as a parameter is set to 0.4. Also, at each training iteration the policy and reward parameters are saved if it reduces the perplexity on the validation set (calculated after running for all the batches of the training dataset) and patience is set to 3 as a stopping criterion before we terminate the training.

Multi-Sentence Knowledge Selection in Open-Domain Dialogue

Mihail Eric, Nicole Chartier, Behnam Hedayatnia, Karthik Gopalakrishnan, Pankaj Rajan, Yang Liu and Dilek Hakkani-Tur
{behnam,karthgop,yangliud,hakkanit}@amazon.com
Amazon Alexa AI

Abstract

Incorporating external knowledge sources effectively in conversations is a longstanding problem in open-domain dialogue research. The existing literature on open-domain knowledge selection is limited and makes certain brittle assumptions on knowledge sources to simplify the overall task (Dinan et al., 2019), such as the existence of a single relevant knowledge sentence per context. In this work, we evaluate the existing state of open-domain conversation knowledge selection, showing where the existing methodologies regarding data and evaluation are flawed. We then improve on them by proposing a new framework for collecting relevant knowledge, and create an augmented dataset based on the Wizard of Wikipedia (WOW) corpus, which we call WOW++. WOW++ averages 8 relevant knowledge sentences per dialogue context, embracing the inherent ambiguity of open-domain dialogue knowledge selection. We then benchmark various knowledge ranking algorithms on this augmented dataset with both intrinsic evaluation and extrinsic measures of response quality, showing that neural rerankers that use WOW++ can outperform rankers trained on standard datasets.

1 Introduction

One of the key components needed to enable robust and engaging open-domain conversational systems is the ability to select and integrate relevant knowledge from diverse knowledge sources. This knowledge selection setting is complex for a number of reasons: 1) relevant knowledge is highly dependent on conversational context, and requires understanding dialogue history and evolving user requirements at a turn-by-turn granularity, 2) because conversations are not topic-constrained, systems may need to pool knowledge from a theoretically boundless number of sources (i.e. the entire internet), and 3) knowledge may be represented in structured, semi-structured, and unstructured formats making it difficult to extract all the information needed for a conversation.

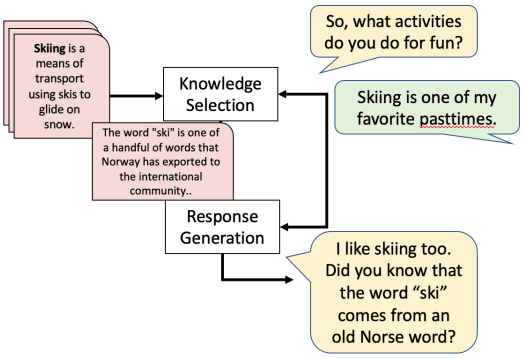


Figure 1: Overview of an end-to-end knowledge ranking and response generation system

Recently there has been increasing interest in the problem of knowledge selection for open-domain dialogue (Gabriel et al., 2020; Dinan et al., 2019; Kim et al., 2020; Gopalakrishnan et al., 2019). There have been numerous efforts to build standardized benchmark datasets whereby open-domain conversations are collected with crowdworkers who are given access to some closed set of knowledge sources such as a subset of Wikipedia or news articles (Dinan et al., 2019; Gopalakrishnan et al., 2019). These datasets suffer from a number of limitations including the fact that they either do not explicitly annotate relevant knowledge sentences per turn in the conversation or make the strict assumption that only a single utterance from a knowledge source is relevant. Without sufficient information about relevant knowledge for a given context, it is difficult to train data-driven models on the isolated problem of knowledge selection.

In this work, we conduct a thorough analysis of the knowledge selection problem through the lens of the Wizard of Wikipedia (WOW) dataset, one

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 76–86,
Aberdeen, Scotland, UK, 20-24 September 2021.

of the standard knowledge-grounded open-domain dialogue benchmarks (Dinan et al., 2019). Our analysis qualitatively and quantitatively demonstrates that the strict one-knowledge-sentence-for-one-context assumption in the data is unreasonable, leading to a lack of meaningful interannotator agreement scores.

We then build on this result and relax this assumption, allowing for multiple knowledge snippets per context. We introduce a new continuously-scaled notion of relevance called *wisdom-of-the-crowd-relevance* (WOC) and use this measure to reannotate about 800 dialog contexts from the original WOW corpus with relevant knowledge. This is done by taking a dialogue from WOW and extracting a subdialogue at a random turn in the dialogue. Our augmented WOW corpus, which we call WOW++, averages 8 knowledge sentences per dialogue turn, and demonstrates significantly more knowledge diversity. Using WOW++, we then benchmark a number of different knowledge ranking algorithms using both standard information retrieval automatic measures as well as extrinsic human evaluation on generated responses. Our results indicate that neural rerankers using WOW++ are able to outperform other algorithms such as traditional IR baselines and neural models trained using the original WOW data.

2 Related Work

In recent years, knowledge selection in open-domain dialogue systems has seen a tremendous surge in interest as the community has recognized the utility of having these abilities in conversational systems (Ram et al., 2018; Khatri et al., 2018; Gabriel et al., 2020).

In one line of work, a number of industry research groups have demonstrated that large quantities of chat data coupled with the latest high-capacity Transformer-based models can produce particularly engaging and convincing conversational experiences (Adiwardana et al., 2020; Roller et al., 2020). While these models produce impressive outputs, they consciously shirk any explicit knowledge-selection mechanisms. Any knowledgeable appearance in their outputs tends to be a consequence of facts memorized in training data (Lux et al., 2020). In addition, the models have a tendency to generate facts that may be factually inaccurate, referred to as *factual hallucination*.

Knowledge selection in open-domain systems took a tremendous leap forward with the introduction of standard datasets of knowledge-grounded dialogues. The work of (Zhou et al., 2018; Dinan et al., 2019; Gopalakrishnan et al., 2019) produced such corpora with upwards of 10K dialogues and up to 10s of dialogue turns leveraging knowledge from diverse sources such as Wikipedia, and the Washington Post. While certainly a step forward, these datasets introduced some unreasonable data constraints that aren't apt to the knowledge setting such as either no explicitly-annotated knowledge snippets or only a single one, making training of robust knowledge selection systems very difficult.

Since the introduction of these corpora, numerous groups have tackled the knowledge selection problem from different angles. For example, Kim et al. (2020) developed a sequential latent variable model to help address ambiguity in knowledge sources in the WOW context. Zhao et al. (2020) developed models that dealt with low-resource settings with general representations learned from ungrounded dialogues but finetuning done with small numbers of domain-specific training examples. Tuan et al. (2020) recognized that even with external knowledge sources, there may be a knowledge gap that can be filled in real-time using unsupervised local knowledge. While these works created modeling improvements on existing datasets, there has still not been a study investigating how well-formed our existing datasets are.

3 WOW++

The WOW++ dataset we describe below is an augmented dataset based on the Wizard of Wikipedia (WOW) corpus (Dinan et al., 2019). The WOW corpus consists of 22,311 dialogues containing 201,999 turns. The dialogues are comprised of two interlocutors who engage in chit chat on a given topic where one interlocutor is a knowledgeable expert in the topic. The expert, or wizard, is provided access to knowledge snippets from Wikipedia that are potentially relevant to a given conversation topic, asked to select one of the knowledge snippets and utilize the information from the knowledge snippet in their response. Thus, for a given wizard utterance only a single knowledge snippet is selected from a set of potentially relevant knowledge snippets. This selected snippet is considered to be the ground-truth, referred to as the *gold* sentence, and is referred to as such throughout this paper.

HUMAN: [...] Do you know any famous guitar players, or have a favorite guitarist that you listen to?

ORIGINAL WOW GROUND-TRUTH KNOWLEDGE SNIPPET:

- Eric Patrick Clapton (born 1945) is an English rock and blues guitarist singer and songwriter.

ALTERNATIVE RELEVANT KNOWLEDGE SNIPPETS:

- George Buddy Guy (born July 30 1936) is an American blues guitarist and singer.
- Doyle Bramhall II (born 24 December 1968) is an American musician, producer, guitarist, and songwriter known for his work with Eric Clapton, Roger Waters, and many others.
- John Mayall OBE (born 29 November 1933) is an English blues singer, guitarist, organist, and songwriter whose musical career spans over fifty years.

Figure 2: Example dialogue in WOW with ground-truth and alternative relevant knowledge snippets.

3.1 Data Motivation: Limitations of One Knowledge

In order to address the well-formedness of existing datasets, we identify two inter-related aspects of knowledge selection in open-domain dialogues: the potential for multiple knowledge snippets to be relevant for a given response, and the subjective nature of relevance selection. Figure 2 exemplifies how multiple knowledge snippets can be equally relevant for a specific question. All knowledge snippets, both the ground-truth snippet and the alternatives, come from Wikipedia. In this example, the response and relevant knowledge that could be leveraged in the response, is open to any guitarist. While the original WOW corpus identifies the knowledge snippet containing information about Eric Clapton as the ground-truth one (i.e. the *gold* sentence) (Dinan et al., 2019), the alternative relevant knowledge snippets are equally relevant. There is nothing inherently more relevant with the Eric Clapton knowledge snippet than these alternatives. The choice, then, for one being *the single relevant* knowledge snippet, would seem to be a reflection of personal preference of the annotator rather than objective relevance.

Annotator subjectivity is not only reflected in questions, but also in open-ended, general statements. Figure 3 depicts this scenario, in which the human's turn leaves the response by the assistant open: there is no direct or leading question to limit the scope of the conversation. In this context, it would be just as reasonable for the system's next turn to leverage the ground-truth knowledge snippet provided by the WOW dataset as it would to leverage the alternative ones shown.

SYSTEM: I work in an animal shelter where abandoned, lost, or stray animals are tended to and rehabbed!

HUMAN: That is good work, I salute you for what you do, that helps a lot of animals out!

ORIGINAL WOW GROUND-TRUTH KNOWLEDGE SNIPPET:

- While no-kill shelters exist, it is sometimes policy to euthanize sick animals and any animal that is not claimed quickly enough by a previous or new owner.

ALTERNATIVE RELEVANT KNOWLEDGE SNIPPETS:

- Critics believe the new term animal shelter is generally a euphemism for the older term pound.
- A no-kill or limited admit shelter is a shelter that saves healthy, treatable, and rehabilitatable animals.
- As a benchmark at least 90% of the animals entering the shelter are expected to be saved.

Figure 3: Example dialogue in WOW with ground-truth and alternative relevant knowledge snippets.

In these cases, the choice of one knowledge sentence over all others reflects a subjective choice by the annotators, who are influenced by their own preconceived notions and expectations of the trajectory of the conversation. As such, although limiting relevance selection to a single knowledge sentence may be beneficial to simplify the annotation task, it does so at the expense of variation that is inherent and expected in open-domain dialogue. Further, the selection of one knowledge snippet in instances where many are relevant reflects annotator preferences, creating an issue of reproducibility.

3.2 Subjectivity of Knowledge Selection

In order to address the feasibility of an annotation task in which multiple knowledge sentences are selected, we conduct a pilot study, adapting the methods used in the WOW study (Dinan et al., 2019). We use a static evaluation that allows annotators to select multiple relevant knowledge snippets for only the final turn of a dialogue. This pilot study includes 40 in-house annotators selecting knowledge snippets for 20 dialogues from the WOW corpus. Annotators are provided with the dialogue and approximately 30 potentially relevant knowledge snippets, including the WOW-identified ground-truth knowledge snippet (Dinan et al., 2019), also from the WOW corpus. These 30 snippets come from, on average, 5 different Wikipedia articles. Annotators are instructed to select all knowledge snippets that could be used in the following turn to craft a relevant response, using the dialogue context to help assess relevance, and that a relevant knowledge sentence should not change the topic of the conversation. Figure 8 in Appendix shows the annotation instruction and interface. Please

note that with this method, a single turn for each dialogue is annotated for knowledge selection.

We find that within the 610 snippets in the pilot annotation, 177 were not selected by a single annotator, i.e. not relevant and only 7 knowledge snippets were selected by all annotators as relevant. Further, we find that only 1 of the 20 gold sentences is selected by all annotators as relevant, the average proportion of annotators that select a gold sentence as relevant is 0.77.

We first assess the quality of the pilot data by computing Fleiss' Kappa to measure inter-annotator agreement (IAA). We compute IAA for relevance selection in the pilot dataset (610 knowledge snippets, assessed by 40 annotators); agreement is moderate ($\kappa = 0.477$, $p < 0.01$) (Landis and Koch, 1977). This finding is not surprising, as we outlined in the previous section that we suspected there would be subjective reactions to knowledge snippets. Next, we explore whether IAA would increase if only the WOW gold snippets are assessed for agreement. To do this, we create a subset of the pilot dataset, consisting of only the relevance assessments by our annotators of the WOW ground-truth knowledge snippets (Dinan et al., 2019) and then compute Fleiss' κ on the subset (20 knowledge snippets, assessed by 40 annotators). IAA for the ground-truth sentences is poor ($\kappa = 0.06$, $p < 0.01$) (Landis and Koch, 1977). Finally, we examine whether low IAA was a reflection of some annotators' understanding of and ability to complete the task. We would expect that if the quality of some annotators' work were subpar, IAA should increase if we find a better subset of annotators. To assess this, we create 100 random samples of 10 annotators from the set of 40 and computed Fleiss' κ for each sample. Agreement in these samples of 10 ranges from fair ($\kappa = 0.40$, $p < 0.01$) to moderate ($\kappa = 0.59$, $p < 0.01$). These results demonstrate that we are unable to create a subset of 10 annotators that agree on relevance selection.

Although low IAA can be an indication of unreliable data (Krippendorff, 1980), it can also be an indication that the task is subjective (Salminen et al., 2018). We argue that low IAA in this context is a result of the inherent subjectivity of the knowledge selection task. Rather than clear and mutually exclusive categories, the notion of relevance in this context has fuzzy boundaries and can be dependent on the individual making the assessment. While there will be some agreement on relevance, it should not be assumed that relevance is an agreed-upon concept. Thus, rather than relying on absolute agreement among all annotators for relevance, we suggest that the notion of relevance be considered on a continuum.

3.3 Data Methodology: Wisdom-of-the-Crowd Relevance

Due to the limitations outlined above, we propose that knowledge selection be handled by appealing to the crowd. Using the same data collection approach as outlined in the pilot study, we conduct a knowledge selection task in which 798 dialogues and corresponding knowledge sentences were randomly sampled from the test seen (198), test unseen (200), and train (400) datasets in the original WOW corpus (Dinan et al., 2019). In order to make the task reasonable for annotators, we create 80 tasks on a crowd-source worker platform in which annotators are presented 10 randomized dialogues. As described above, for each dialogue a single turn is annotated for knowledge selection. 10 annotators assess knowledge snippets for these 798 dialogues, with an average of 30 knowledge snippets per dialogue. In order to mitigate low-quality annotations, we include two filter metrics in the task.

Determining the threshold for relevance, or *wisdom-of-the-crowd-relevance*, consists of a mixed approach of relevance vote distribution and manual inspection. The relevance vote for each knowledge sentence represents the proportion of annotators that selected a given knowledge sentence as relevant. We order the knowledge snippets from all dialogues by this relevance vote and use the third quartile of the vote distribution as the cut-off for relevance, resulting in a relevance threshold of 0.6. A sample of relevant knowledge sentences is manually inspected to ensure the quality and accuracy of the *wisdom-of-the-crowd-relevance*. This approach to relevance accounts for variation due to inherent subjectivity while limiting noise expected from human evaluation.

The *wisdom-of-the-crowd-relevance* scoring results in an average of 8 selected knowledge snippets per turn. Table 1 provides a summary of relevant knowledge snippets in the WOW++ dataset. Figure 4 shows the distribution of relevant knowledge for the WOW++ dataset. Only 5% of the dialogues in the dataset contain a single relevant knowledge snippet. These results suggest that, for the majority of dialogues, more than one knowledge sentence is

relevant, and more importantly, that only a single knowledge sentence being relevant is the *exception*, not the norm.

Number of Dialogues	798
Average Relevant KS per Turn	8
Turns with no Relevant KS	39
Turns with 1 Relevant KS	41
Turns with >1 Relevant KS	718

Table 1: Counts of relevant knowledge snippets (KS) in WOW++

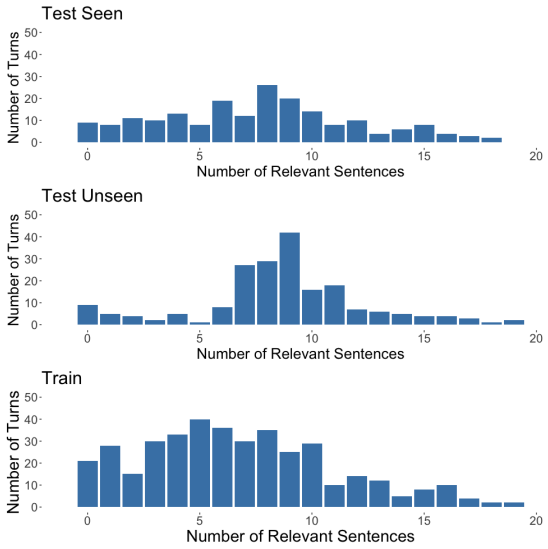


Figure 4: Histograms of the frequency counts for different numbers of relevant snippets per turn annotated from the test seen, test unseen, and train data.

To better understand the conversational contexts that have multiple relevant knowledge snippets, we examine a sample of the data to categorize the users' last utterance. Table 2 depicts the three broad categories: personal questions; factual questions; and general statements, and their distribution across our sample. Overall, the most frequent type is when the user's last utterance was a general statement. We then examined whether the multiple knowledge snippets came from the same topic (i.e. Wikipedia article) or were spread across the five different topics presented. Approximately 50% of the relevant knowledge comes from a single topic for all three categories. For both general statements and factual questions, the remaining 50% of knowledge is spread across 2 - 5 topics. For personal questions, 40% of the knowledge comes from only two topics, and the final 10% comes from 5 topics.

Finally, we examined the original WOW gold

Category	Example	%
Personal	I love to play football, do you?	10
Factual	What else do you know about rap?	33
General	Soap operas seem so bad to me.	57

Table 2: Types of last user utterances and percent of occurrence in our dataset.

HUMAN: I've had my heart broken a few times, how about you?

ORIGINAL WOW GROUND-TRUTH KNOWLEDGE SNIPPET:

- It is the day before Valentine's Day and Phil (Ty Burrell) and Claire (Julie Bowen) decide to bring back their alter egos, Clive Bixby and Juliana

ALTERNATIVE RELEVANT KNOWLEDGE SNIPPETS:

- The concept is cross-cultural, often cited with reference to a desired or lost lover, and dates back to at least 3000 years.
- The emotional pain of a broken heart is believed to be part of the survival instinct.
- The concept is believed to be universal, with many cultures using the same words to describe both physical pain and the feelings associated with relationship loss.

Figure 5: Example dialogue where WOW original "gold" is not relevant, but alternative snippets are.

snippet in relation to our multiple knowledge selection method. After removing dialogues where gold snippets were not included among the 30 candidates, there are a total of 697 conversations where the gold snippet is presented among the potentially relevant knowledge sources. Of those, there are 10 dialogues where the gold snippet is the only relevant knowledge snippet. However, there are 160 dialogues where the gold snippet does *not* meet the relevance threshold. Although the gold snippet is not relevant in these 160 conversations, 136 have at least one alternative knowledge snippet selected as relevant. We inspect these instances and find that, in general, it is due to noise in the original WOW dataset. The dialogue in Figure 5 exemplifies this noise. Reading the WOW gold snippet, it is not clear how the knowledge in that snippet could be leveraged accurately to craft a relevant response to the question about heartbreak. This suggests that while a single person was able to craft a relevant response from this snippet, in general, it would not be seen as relevant. In other words, although the gold snippet is not relevant in these conversations, other knowledge snippets are, suggesting that there are knowledge snippets that are more "gold" than the WOW gold snippet.

By allowing for multiple relevant knowledge and introducing *wisdom-of-the-crowd-relevance* scor-

ing, we have produced a robust augmented dataset that embraces the variation present in open-domain knowledge selection. We have demonstrated the assumption of one-knowledge-snippet-per-context needs to be re-assessed, as our data suggests that a single relevant knowledge snippet may not be reasonable nor replicable. Not only does this method help to mitigate some noise present in the original WOW dataset (Dinan et al., 2019), but we expect that it will be more fruitful when incorporating knowledge sources beyond Wikipedia.

4 Method

We evaluate WOW++ using two different regimes to see the effect of the augmented knowledge snippets. The first is intrinsic evaluation for the knowledge reranking or selection task using automatic measures. The second is extrinsic evaluation where we provide a selected knowledge candidate to a response generation model and perform human annotation of system generated responses.

4.1 Knowledge Selection

This intrinsic evaluation is intended to assess the performance of different models within the context of an isolated knowledge selection module. More formally, assume we are given a dialogue context consisting of m turns represented as $D = \{T_1, T_2, ..., T_m\}$, and for the last turn, a list of knowledge sentence candidates are provided, $C = \{s_1, s_2, ..., s_n\}$, the system is asked to assess the relevance of these candidates and generate a ranked list, i.e., $f(C, D) = \{s_{i_1}, s_{i_2}, ..., s_{i_m}\}$.

We first evaluate using unsupervised sentence similarity scores for knowledge sentence relevance ranking, with scores calculated using traditional Tf-idf method or large pretrained models.

- **Tf-IDF**: Here we separately encode the dialogue context and each knowledge sentence from the set C using term-frequency (TF) inverse-document-frequency (IDF) weights and then return the top 10 sentences that are most similar to the context vector using the cosine similarity metric. The IDF weights are learned from the full train set of the new WOW++ dataset by treating each sentence of the knowledge passages and dialogue turns as documents.
- **Static RoBERTa Retriever**: Here we encode the context and knowledge sentences using an off-the-shelf non-finetuned RoBERTa base model (Liu et al., 2019). We then compute cosine similarity and return the top-ranked sentences.
- **MLM tuned RoBERTa Retriever**: Here we use the same scheme as in the *Static RoBERTa* case except that the RoBERTa weights are first finetuned using a masked language modeling (MLM) objective on the WOW train set. This is analogous to the self-supervised pretraining described in (Mehri et al., 2020).

In addition, we propose to use supervised learning to train knowledge rerankers based on the pretrained language models, RoBERTa. We fine tune the RoBERTa base model by scoring an input sequence consisting of a dialogue context and a candidate knowledge sentence using a binary log-loss objective. This trained RoBERTa reranker outputs a binary probability of relevance for the context and candidate pair, which is used to rank the candidates. In the following, we describe different training configurations in order to examine their effect on knowledge selection performance. For all the RoBERTa models, we use a maximum length of 256 tokens for the concatenated dialog context and knowledge candidate.

- **Training on WOW++**: we finetune a RoBERTa base model on WOW++ data. In training, a candidate knowledge sentence is considered positive if it exceeds our WOC relevance threshold.
- **Training on WOW-original**: the original WOW train set is used as the labelled data. Here we leverage one positive candidate (the gold knowledge snippet) and one negative candidate (sampled from the remaining knowledge) per dialogue context. Note that this scheme has about 120K examples, roughly an order of magnitude more data for training compared to only using WOW++.
- **Training on WOW-original-subset**: here we use the same dialogs as used in WOW++, however, rather than using the new annotated snippets from WOW++, we use the gold knowledge snippet from the original WOW dataset corresponding to each context of the 400 training dialogues. Since this only introduces a single positive snippet per context, we additionally sample enough negative candidates from the available knowledge in each

		MRR@1	MRR@5	MAP@5	MAP@10	NDCG@5	NDCG@10
TF-IDF		0.66/0.74	0.76/0.84	0.56/0.65	0.57/0.63	0.80/0.87	0.81/0.86
Roberta Retriever	Static RoBERTa	0.51/0.58	0.66/0.68	0.39/0.45	0.36/0.38	0.70/0.73	0.72/0.75
	Finetuned RoBERTa	0.68/0.71	0.78/0.82	0.55/0.64	0.53/0.59	0.81/0.85	0.81/0.85
Roberta classifier	WOW ++	**0.75/0.84**	**0.83/0.89**	**0.67/0.75**	**0.67/0.73**	**0.86/0.90**	**0.86/0.90**
	WOW-original	0.65/0.71	0.76/0.81	0.56/0.63	0.55/0.60	0.79/0.85	0.81/0.86
	WOW-original-subset	0.30/0.45	0.46/0.59	0.25/0.37	0.27/0.37	0.52/0.65	0.58/0.68
	WOW-original + WOW++	0.71/0.82	0.80/0.88	0.62/0.70	0.63/0.67	0.82/0.89	0.83/0.89

Table 3: Knowledge selection: automatic evaluation results by data split (Test Unseen/Test Seen).

context so that the total samples used for training match the number used in the case when training with WOW++ data.

- **Training on WOW-original + WOW++**: Here the training data contains both the WOW++ and the original WOW train set.

4.2 Response Generation

The extrinsic evaluation is intended to evaluate the knowledge selection module in the context of a fully-integrated dialog response generation system, thereby giving us a better understanding of end-to-end performance. We seek to understand the effect of different quality knowledge sentences on the downstream task of response generation.

Here, we first finetune the language modeling head of a GPT2 medium-model (Radford et al., 2019) on the original WOW corpus, using a similar setup as (Wolf et al., 2019) where the ground-truth knowledge and response are used for teacher-forcing training. The context is truncated to 64 tokens. During inference, we take the top ranked sentence from a knowledge reranking model, use it along with the dialog context as a concatenated input to generate a response.

5 Experimental Results

5.1 Knowledge Selection Results

For knowledge selection, we evaluate our models using a number of standard information retrieval metrics: MAP (mean average precision) and NDCG (normalized discounted cumulative gain) for the 5 and 10 candidate decision thresholds, and MRR (mean reciprocal rank) for the 1 and 5 candidate decision thresholds.

Table 3 shows the results for the knowledge selection methods presented in Section 4. First for the similarity-based methods, we see the traditional TF-IDF measure has strong performance. This may also speak to the manner in which the WOW data collection was done whereby crowdworkers could have optimized for snippets with high degrees of overlap with the dialogue context rather than necessarily those with the highest levels of semantic relevance. This is certainly an artifact of the original WOW data collection process where candidate articles were chosen via their TF-IDF weighted n-gram scores with the dialogue context. Using the static RoBERTa model to computer similarity does not perform as well as the TF-IDF metrics, again, partly because of the reason discussed above. Adapting the RoBERTa model to the task domain using the WOW data in an unsupervised fashion via MLM loss improves the performance significantly over the static models. This is not surprising since the model is further trained on the domain data, resulting in better representations for words and sequences for similarity measures.

Regarding the RoBERTa classifier ranking approaches, results show that the model trained using the WOW++ training data achieves the best performance, also outperforming the similarity-based unsupervised methods. This shows that neural models for knowledge selection benefit from supervised training. Among different configurations for the RoBERTa classifiers, we can see that training on WOW++ is the most effective across different metrics. Given that the training data size is about the same between WOW++ and WOW-original-subset, the performance gap can be explained by the fact that only a single positive snippet was provided in the latter, whereas multiple positive knowledge sentences are used in WOW++, which is a matched condition for our test setup. We also varied the setting for WOW-original-subset by only including one negative example, i.e., total 800 classification training instances. This improved the performance slightly, i.e., MRR@1 is 0.35/0.46, but still a large gap with using WOW++ training data. Comparing to WOW-original, though WOW++ is much smaller, the model was able to better leverage the information from multiple relevant knowledge snippets and learn the optimal ordering of knowledge

candidates. The last row in the table again shows that matched training is important. When adding WOW-original to WOW++ for training, the results are not as competitive as just using WOW++ data.

Between the seen and unseen split, the results are generally as expected. However, for the unsupervised methods, we would expect smaller difference since there is no notion of 'seen' topics. One reason for this is that the IDF is in fact learned from the training set, and finetuned RoBERTa retriever also has in-domain unsupervised pre-training. We will investigate the effect of topics further in our future work.

5.2 Response Generation Results

For extrinsic evaluation, we perform human evaluation of the responses generated using different knowledge selection methods. We also experimented with computing automatic evaluation metrics such as ROUGE with respect to the human responses in the original WOW dataset but found the results quite low. This is to be expected given the sensitivity of generated responses to the knowledge we provide in our systems.

First we evaluate the effect of different ground truth, *Wisdom-of-Crowd* and *WoW Original Gold*, on response generation. The *Wisdom-of-Crowd* setting uses the most relevant knowledge sentence according to the WOC scores from WOW++, whereas the *WoW Original Gold* setting uses the gold relevant snippet from the original WOW dataset. We randomly sampled 100 dialogue contexts for human evaluation. Each dialogue context coupled with the system output is provided to an annotator who is asked to evaluate the output according to appropriateness and informativeness. The responses were evaluated by 2 expert annotators on an ordinal 0-2 scale for each metric. Results are provided in Table 4. It is clear that the single ground truth in the original WOW data is not as good as the WOC scoring scheme for picking good knowledge snippets to feed to the downstream response generator.

Knowledge used	**Appropriate**	**Informative**
WOW Original Gold	1.33/1.60	1.00/1.18
Wisdom-of-Crowd	**1.34/1.70**	1.25/**1.29**

Table 4: Response generation: human evaluation results of responses when ground truth knowledge is provided to the NRG model (Test Unseen/Test seen).

Next we compare the responses generated using different automatic knowledge selection approaches. Based on results from Table 3, we just ran this human evaluation for two methods, TF-IDF and RoBERTa reranker. For this evaluation, since providing absolute scores is more subjective, we performed a preference test by asking annotators to choose which response is better between the two candidates, on two dimensions: appropriate and informative. Results in Table 5 show that consistent with the intrinsic knowledge selection results, the RoBERTa model trained on the WOW++ data performs better, showing it is more able to provide relevant knowledge to be used by the downstream response generator. One problem we found with the TF-IDF method is that it may select a knowledge sentence that repeats information in the dialog context. This is not surprising since it is heavily relying on lexical overlap, whereas the supervised RoBERTa reranker has learned about both relevance and repetition during training. Examples in Appendix show this issue for TF-IDF.

Knowledge used	**Appropriate**	**Informative**
TF-IDF	21.5%	22%
RoBERTa	49.5%	47.5%

Table 5: Response generation using different knowledge selection method: TF-IDF vs. RoBERTa. Results show the percentage of the method chosen as the preferred one for that dimension.

6 Conclusion

In this work, we demonstrated that knowledge selection is an intrinsically ambiguous task for open-domain conversations, which necessitates improved notions of relevance in our benchmark datasets. We used this insight to propose a new measure for knowledge sentence relevance called *Wisdom-of-the-Crowd* relevance. Using this measure, we annotated a new collection of dialogues with relevant knowledge called WOW++ (it will be released publicly). We then evaluated a number of knowledge selection algorithms on our new dataset using both intrinsic and extrinsic metrics, and demonstrate that neural rankers trained leveraging WOW++ can outperform traditional knowledge selection algorithms. It is worth noting, however, that annotating a knowledge selection dataset with all relevant snippets as we have done for WOW++ is a time-intensive task that may be expensive to scale up. Future work should investigate how to develop more low-resource rankers or how to bootstrap from a high quality seed dataset like WOW++ to a larger corpus.

References

D. Adiwardana, Minh-Thang Luong, D. So, J. Hall, Noah Fiedel, R. Thoppilan, Z. Yang, Apoorv Kulshreshtha, G. Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a human-like open-domain chatbot. *ArXiv*, abs/2001.09977.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, M. Auli, and J. Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. *ArXiv*, abs/1811.01241.

Raefer Gabriel, Yang Liu, Mihail Eric Anna Gottardi, Anju Khatri, Anjali Chadha, Qinlang Chen, Behnam Hedayatnia, Pankaj Rajan, Ali Binici, Shui Hu, Karthik Gopalakrishnan, Seokhwan Kim, Lauren Stubel, Arindam Mandal, , and Dilek Hakkani-Tür. 2020. Further advances in open domain dialog systems in the third alexa prize socialbot grand challenge. Alexa Prize Proceedings.

Karthik Gopalakrishnan, Behnam Hedayatnia, Q. Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, R. Gabriel, and D. Hakkani-Tur. 2019. Topical-chat: Towards knowledge-grounded open-domain conversations. In *INTERSPEECH*.

C. Khatri, Behnam Hedayatnia, Anu Venkatesh, J. Nunn, Yi Pan, Q. Liu, H. Song, Anna Gottardi, Sanjeev Kwatra, S. Pancholi, Ming Cheng, Qinglang Chen, Lauren Stubel, Karthik Gopalakrishnan, Kate Bland, R. Gabriel, A. Mandal, Dilek Z. Hakkani-Tür, Gene Hwang, Nate Michel, E. King, and R. Prasad. 2018. Advancing the state of the art in open domain dialog systems through the alexa prize. *ArXiv*, abs/1812.10757.

Byeongchang Kim, Jaewoo Ahn, and G. Kim. 2020. Sequential latent knowledge selection for knowledge-grounded dialogue. *ArXiv*, abs/2002.07510.

Klaus Krippendorff. 1980. Validity in content analysis.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Klaus-Michael Lux, Maya Sappelli, and Martha Larson. 2020. Truth or error? towards systematic analysis of factual errors in abstractive summaries. In *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 1–10, Online. Association for Computational Linguistics.

Shikib Mehri, M. Eric, and D. Hakkani-Tur. 2020. Dialoglue: A natural language understanding benchmark for task-oriented dialogue. *ArXiv*, abs/2009.13570.

A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

A. Ram, Rohit Prasad, C. Khatri, Anu Venkatesh, R. Gabriel, Q. Liu, J. Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, E. King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrue. 2018. Conversational ai: The science behind the alexa prize. *ArXiv*, abs/1801.03604.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, M. Williamson, Yinhan Liu, J. Xu, Myle Ott, Kurt Shuster, Eric Michael Smith, Y.-Lan Boureau, and J. Weston. 2020. Recipes for building an open-domain chatbot. *ArXiv*, abs/2004.13637.

Joni O Salminen, Hind A Al-Merekhi, Partha Dey, and Bernard J Jansen. 2018. Inter-rater agreement for social computing studies. In *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 80–87. IEEE.

Yi-Lin Tuan, Wei Wei, and William Yang Wang. 2020. Unsupervised injection of knowledge into dialogue generation via language models. *ArXiv*, abs/2004.14614.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *ArXiv*, abs/1901.08149.

Xueliang Zhao, Wei Wu, Chongyang Tao, Can Xu, Dongyan Zhao, and Rui Yan. 2020. Low-resource knowledge-grounded dialogue generation. *ArXiv*, abs/2002.10348.

Kangyan Zhou, Shrimai Prabhumoye, and A. Black. 2018. A dataset for document grounded conversations. *ArXiv*, abs/1809.07358.

A: What do you think about the band nickelback?
B: Nickelback is a Canadian rock band formed in 1995 that is not very good.

TF-IDF knowledge:

- Nickelback <knowledge_separator> Nickelback is a Canadian rock band formed in 1995 in Hanna, Alberta, Canada.
- Nickelback <knowledge_separator> The band signed with Roadrunner Records in 1999 and re-released their once-independent album "The State".
- Nickelback <knowledge_separator> Nickelback is one of the most commercially successful Canadian groups, having sold more than 50 million albums worldwide and ranking as the eleventh best-selling music act, and the second best-selling foreign act in the U.S. of the 2000s, behind The Beatles.

RoBERTa knowledge:

- Nickelback <knowledge_separator> The band is composed of guitarist and lead vocalist Chad Kroeger, guitarist, keyboardist and backing vocalist Ryan Peake, bassist Mike Kroeger, and drummer Daniel Adair.
- Nickelback <knowledge_separator> The band achieved commercial success with the release of their 2000 album \"The State\" and then they achieved mainstream success with the release of their 2001 album "Silver Side Up".
- Nickelback <knowledge_separator> Nickelback is one of the most commercially successful Canadian groups, having sold more than 50 million albums worldwide and ranking as the eleventh best-selling music act, and the second best-selling foreign act in the U.S. of the 2000s, behind The Beatles.

Figure 6: Example dialogues showing the top 3 ranked knowledge snippets for both the *Tf-IDF* and *RoBERTa Reranker* models. Note how the *RoBERTa Reranker* tends to select knowledge that is more semantically coherent with the most recent dialogue context. By comparison, the *TF-IDF* model only focuses on snippets with high lexical overlap, resulting in repeated information.

Human: i love eating kale on sushi, Kale or leaf cabbage are certain cultivars of cabbage

System responses using different knowledge:

TF-IDF:
I love kale! I can't believe it's edible leaves.

Roberta ranker:
Kale was brought to North America by the colonists in the 16th century.

WOW original gold:
I love kale. It is actually a leaf cabbage.

Wisdom-of-the-crowd:
Kale was brought to North America by the colonists in the 16th century.

Figure 7: Example dialogue with corresponding responses. Note that both the *TF-IDF* and *WOW original gold* responses repeat information that was previously given by the user's turn.

INSTRUCTIONS: Read the conversation below between a human and a virtual assistant. The assistant should try and craft their next response using knowledge from the snippets provided. To give the assistant different options for useful knowledge, your task is to read the knowledge snippets and choose **each** snippet that the assistant could reasonably use in their next response. You should choose as many knowledge snippets as you think are relevant.

CONVERSATION:

HUMAN: I love dogs... my favorite is the Golden Retriever!
ASSISTANT: Goldens are easy to train and love water.
HUMAN: Do you know how smart they are compared to other breeds?
ASSISTANT: I assume they are quite smart as they are trained to be search and rescue, hunting dogs, and guides for the blind.
HUMAN: Wow, so they're working dogs. Are they good with families?
ASSISTANT: Their gentle temperament makes them the third-most popular family dog breed.
HUMAN: The third most popular, do you know what any other popular family breeds are? I'm trying to keep an open mind.

KNOWLEDGE SNIPPETS:

GOLDEN RETRIEVER:
- ☐ Golden Retriever is a large-sized breed of dog bred as gun dogs to retrieve shot waterfowl, such as ducks and upland game birds during hunting and shooting parties and were named 'retriever' because of their ability to retrieve shot game undamaged.
- ☐ The Golden Retriever is popular as a disability assistance dog such as being a guide dog for the blind and a hearing dog for the deaf.

BEAGLE:
- ☐ The beagle is a breed of small hound that is similar in appearance to the much larger foxhound.
- ☐ It is a popular pet due to its size, good temper, and lack of inherited health problems.

TALBOT (DOG)
- ☐ The Talbot was a type of white hunting dog.
- ☐ It is now extinct and has been credited with being an ancestor of the modern beagle and bloodhound.

Figure 8: Adapted version of knowledge selection task presented to crowdworkers.

Self-Training for Compositional Neural NLG in Task-Oriented Dialogue

Xintong Li, Symon Jory Stevens-Guille, Aleksandre Maskharashvili and Michael White

Department of Linguistics
The Ohio State University
znculee@gmail.com stevensguille.1@osu.edu
maskharashvili.1@osu.edu mwhite@ling.osu.edu

Abstract

Neural approaches to natural language generation in task-oriented dialogue have typically required large amounts of annotated training data to achieve satisfactory performance, especially when generating from compositional inputs. To address this issue, we show that self-training enhanced with constrained decoding yields large gains in data efficiency on a conversational weather dataset that employs compositional meaning representations. In particular, our experiments indicate that self-training with constrained decoding can enable sequence-to-sequence models to achieve satisfactory quality using vanilla decoding with five to ten times less data than with ordinary supervised baseline; moreover, by leveraging pretrained models, data efficiency can be increased further to fifty times. We confirm the main automatic results with human evaluations and show that they extend to an enhanced, compositional version of the E2E dataset. The end result is an approach that makes it possible to achieve acceptable performance on compositional NLG tasks using hundreds rather than tens of thousands of training samples.

1 Introduction

Neural approaches to natural language generation (NLG) have received increasing attention due to their flexibility and end-to-end trainability (Wen et al., 2016; Mei et al., 2016; Dušek and Jurcicek, 2016; Dušek et al., 2019). However, despite using simplistic input meaning representations (MR), most neural models require large quantities of clean annotated training data in order to obtain good performance. As such, the time and expense required to obtain sufficient training data is a significant obstacle to deploying neural NLG models at scale.

To enable richer task-oriented dialogue, Balakrishnan et al. (2019) argue for using compositional, tree-structured MRs that include discourse relations, emphasizing the need for applications to exert control over these relations when generating text. Perhaps not surprisingly, their compositional input MRs further exacerbate annotated data needs. To address this issue, Balakrishnan et al. (2019) introduce a novel constrained decoding technique that nearly always yields correct output even in challenging cases. However, their constrained decoding method incurs a substantial runtime cost, making it too slow to deploy in task-oriented dialogue systems where low latency is a priority. Thus, finding ways to improve data efficiency for training models that perform satisfactorily with vanilla decoding remains an important challenge.

In order to reduce annotated data needs, Kedzie and McKeown (2019) and Qader et al. (2019) propose self-training methods for NLG, though they do not explore self-training for the more challenging case of generating from compositional input representations. Arun et al. (2020) do explore self-training with compositional inputs, but they do not consider constrained decoding. In this paper, we investigate for the first time whether constrained decoding can be used during self-training to enhance data efficiency for compositional neural NLG, since the speed of constrained decoding is much less of a concern during self-training than it is at runtime in dialogue systems. In particular, we adapt and extend He et al.'s (2020) approach to self-training for MT to the setting of neural NLG from compositional MRs, comparing vanilla self-training to self-training enhanced with constrained decoding as well as with reverse model reranking (Shen et al., 2019; Yee et al., 2019), a simpler technique where the n-best outputs of the forward model are reranked using scores from a reverse model. In both cases, the idea is to enhance the quality of the pseudo-annotated texts created during self-training, so that self-training can more successfully avoid entrenching the model's own

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 87–102,
Aberdeen, Scotland, UK, 20-24 September 2021.

Query	Context	MR
When will it snow next?	Reference date: 29th September 2018	[CONTRAST [INFORM [LOCATION [CITY Parker]] [CONDITION_NOT snow] [DATE_TIME [DAY 29] [MONTH September] [YEAR 2018]]] [INFORM [DATE_TIME [DAY 29] [MONTH September] [YEAR 2018]] [LOCATION [CITY Parker]] [PRECIP_CHANCE_SUMMARY very likely] [CONDITION heavy rain showers] [CLOUD_COVERAGE partly cloudy]]]

Annotated Response

[CONTRAST [INFORM [LOCATION [CITY Parker]] is not expecting any [CONDITION_NOT snow]] , but [INFORM [DATE_TIME [COLLOQUIAL today]] there's a [PRECIP_CHANCE_SUMMARY very likely chance] of [CONDITION heavy rain showers] and it'll be [CLOUD_COVERAGE partly cloudy]]]

Table 1: Example compositional MR and annotated response from Balakrishnan et al.'s (2019) conversational weather dataset. In the actual dataset, discourse relations have a DS prefix (e.g., DS_CONTRAST), dialog acts have a DG prefix (e.g, DG_INFORM) and arguments have an ARG prefix (e.g., ARG_CITY); these are elided here for brevity.

mistakes. We show that self-training benefits considerably from both methods, and that constrained decoding yields especially large gains in data efficiency. In particular, our experiments indicate that using constrained decoding during self-training, rather than at runtime, enables standard sequence-to-sequence (seq2seq) models to achieve satisfactory quality with much reduced latency.

Our contributions are two-fold. On Balakrishnan et al.'s (2019) conversational weather dataset, we show that using constrained decoding during self-training and their SEQ2SEQ-TREE model at runtime yields comparable performance with 20% of the annotated data as using the full training set in supervised fashion, and by leveraging pretrained models, annotated data needs can be further reduced to 2%. We then confirm the main automatic metric results with human evaluations and show that they hold for Balakrishnan et al.'s (2019) enhanced version of the E2E dataset (Dušek et al., 2019).

2 Method

Neural NLG seq2seq models aim to generate a natural language text $\mathbf{y} = \langle \mathrm{y}_1, \cdots, \mathrm{y}_{|\mathbf{y}|} \rangle$ from a meaning representation $\mathbf{x} = \langle \mathrm{x}_1, \cdots, \mathrm{x}_{|\mathbf{x}|} \rangle$ by modeling the conditional probability

$$P(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{|\mathbf{y}|} P(\mathrm{y}_i|\mathbf{y}_{<i}, \mathbf{x}), \quad (1)$$

where $\mathbf{y}_{<i} = \langle \mathrm{y}_1, \ldots, \mathrm{y}_{i-1} \rangle$ denotes a prefix of $\mathbf{y}$ with length $i-1$. Usually, the model parameters are learned in supervised fashion from a set of annotated data $\mathcal{L} = \{\mathbf{x}_k, \mathbf{y}_k\}_{k=1}^{|\mathcal{L}|}$.

2.1 Compositional Inputs

Balakrishnan et al. (2019) propose to generate annotated responses from compositional, tree-structured MRs, as shown in Table 1. They demonstrate that compositional MRs offer greater control over the expression of CONTRAST and JUSTIFICATION discourse relations and lead to improvements in semantic correctness in a human evaluation, which they argue is important for conversational systems where external knowledge like user models may inform decisions around contrast, grouping, or justifications (Carenini and Moore, 2006; Walker et al., 2007; White et al., 2010; Demberg et al., 2011). By serializing the trees as shown, it is possible to use standard seq2seq models to effectively accomplish tree-to-tree generation. At runtime, the bracketing tokens can be straightforwardly removed to produce the final outputs.

2.2 Vanilla Self-Training

Hiring annotators to produce large amounts of clean, parallel data is costly, but it is often possible to automatically obtain lots of unlabeled data $\mathcal{U} = \{\mathbf{x}_l\}_{l=1}^{|\mathcal{U}|}$. To take advantage of the large unlabeled data $\mathcal{U}$, we adapt and extend He et al.'s (2020) semi-supervised self-training strategy, which has been successfully applied to MT. As shown in Algorithm 1, vanilla self-training starts from a base

model trained with annotated parallel data $\mathcal{L}$, then (i) iteratively applies the current model to pseudo-label the unlabeled data with its predictions, (ii) trains a new model on the pseudo-labeled data, and (iii) fine-tunes the model on $\mathcal{L}$. Naturally, higher-quality pseudo-labeling can be expected to lead to more effective self-training by helping the model to avoid entrenching its own mistakes; below, we consider two strategies for improving generation during the pseudo-labeling step.

Algorithm 1: Vanilla Self-Training

```
Train a model on L;
repeat
    Pseudo-label the unlabeled data in U;
    Train a model on the pseudo-parallel
      data;
    Fine-tune the model on L;
until convergence or maximum iteration;
```

2.3 Constrained Decoding

Balakrishnan et al. (2019) demonstrate that constrained decoding can enhance the correctness of text generated with seq2seq models. In our experiments, we make use of an enhanced version of their constrained decoding method, both in the pseudo-labeling step of self-training as well as during runtime prediction.

Balakrishnan et al.'s (2019) constrained decoding method begins by scanning the input MR tree to build constraints on coverage and ellipsis.[1] During decoding, the non-terminals in the incrementally generated candidates are checked against the input tree for validity, where an output tree (ignoring terminals) is considered valid if it is isomorphic to the input tree up to sibling order and elided arguments. After each time step of the beam search, invalid candidates are filtered out to prevent hallucinations of tree structure, and closing brackets can only be generated when the non-terminals in the current subtree have all been covered. For example, in decoding a response for the MR in Table 1, if the prediction has followed the annotated response up until *and it'll be*, then a closing bracket cannot be generated at this point because the second INFORM is not complete, and CLOUD_COVERAGE is the only non-terminal that can be validly generated here.

[1] Arguments appearing multiple times in the input MR are only required to appear once in the output.

A problem with this post-filtering method of constrained decoding is that it can end up filtering out all candidates in the beam search, making it impossible for the decoding to proceed forward. To avoid this issue, we instead make use of a pre-filtering constraint. Specifically, rather than checking the non-terminals in $\mathbf{y}_{\leqslant i}$ after generating the next token in each time step i, our pre-filtering method instead determines all non-terminals that can appear as valid next tokens with $\mathbf{y}_{<i}$, then masks out all invalid non-terminals from the vocabulary before the next decoding step (the closing bracket is treated similarly). This ensures that all candidates in the beam are valid.

Another problem with Balakrishnan et al. (2019)'s constrained decoding is that it only constrains the generation of non-terminals. The generated terminals may be inconsistent with their parent argument non-terminals, even when placeholder terminals are used for delexicalized arguments. For example, a placeholder for city name should only be valid to generate inside an [ARG_CITY] argument instead of [ARG_DAY]. This kind of error is not common when the training data is sufficient, but it can severely harm the generation quality in data sparse situations. Therefore, in our enhanced constrained decoding, we constrain the generation of arguments by only nominating correspondingly valid placeholder terminals given a particular parent argument non-terminal.

While constrained decoding ensures the correctness of the partial tree structure and helps avoid inappropriate argument realizations, it does not constrain most terminals (i.e., the words themselves). As such, when the model ends up in a poorly trained part of its distribution, it can still hallucinate terminals; in particular, it can end up stuttering words until the maximum output length is reached, yielding an invalid tree structure. In these failure cases, we replace the output with the result of vanilla decoding, whose text is usually much better.

2.4 Reverse Model Reranking

As an alternative to constrained decoding's hard constraints on non-terminals, we also investigate a soft approach to favoring generated texts that correctly express the input MRs (Shen et al., 2019; Yee et al., 2019). To score the correctness of a generated text (with non-terminals removed), we train a reverse (i.e., parsing) model to generate a mean-

ing representation $\mathbf{x}$ from a natural language text $\mathbf{y}$. Then, following beam search, the n-best generated texts are reranked with the forced decoding perplexity of the reverse model. When using reverse model reranking in self-training, the reverse model is also self-trained as shown in Algorithm 2.

Algorithm 2: Reverse Model Reranking

```
Train forward and reverse models on ℒ;
repeat
    Pseudo-label the unlabeled data in 𝒰
      with reverse model reranking;
    Train forward and reverse models on the
      pseudo-parallel data;
    Fine-tune both models on ℒ;
until convergence or maximum iteration;
```

3 Experiments

3.1 Setup

Datasets We conduct experiments on the publicly available conversational weather and enriched E2E datasets from Balakrishnan et al. (2019), focusing on the more challenging weather dataset. The weather task consists of 25k parallel items for training, and 3k for both validation and test. In the weather task, there are 1.6k unique tokens in the MRs, and 1.3k in the annotated responses. The enriched E2E dataset contains Balakrishnan et al.'s (2019) automatic enhancements to the E2E texts and MRs to include CONTRAST and JUSTIFICATION relations as well as slot-level annotations. The E2E task consists of 42k items for training, and 4.6k for both validation and test. In the E2E task, there are 60 unique tokens in the MRs, and 2.9k in the annotated responses. All the results are reported on the test set in the following experiments.

Unlabeled MR Creation For many NLG applications, unlabeled MRs can be generated in nearly unlimited quantities with a simulator, but unfortunately, we do not have access to the MR simulators for these two datasets. Our workaround is to create unlabeled MRs by modifying the MRs we have in the parallel data. Because there are contextual dependencies in the MRs, it would be hard to get realistic MRs just by sampling elements. Therefore, we instead delete all possible combinations of removable subtrees from the MRs in order to keep the pruned MRs meaningful. The removable subtrees are defined as an unprotected DG_INFORM or ARG that has at least one unprotected sibling, where protected elements are those that are manually identified as establishing context (e.g., ARG_LOCATION) or are children of CONTRAST and JUSTIFICATION relations, which have coherence-related contextual dependencies. In this way, we created 137k unlabeled MRs for weather and 143k MRs for E2E. When training a new model on pseudo-labeled data, we split 3k from each of them as validation data.

Models We report results for the following four kinds of models, where *-n means the method only uses $n\%$ of the parallel data from the full training set (three iterations of self-training were used, unless otherwise specified):

- LBL-n: A seq2seq model (LSTM with attention or BART), which is also the base model for the other methods
- ST-VAN-n: A model trained with vanilla self-training
- ST-CD-n: A model self-trained with constrained decoding for pseudo-labeling
- ST-RMR-n: A model self-trained with reverse model reranking for pseudo-labeling

Metrics We report the automatic metrics listed below on the raw model predictions, which have delexicalized fields (e.g., ARG_CITY). Non-terminal annotations are stripped when calculating BLEU-4 and auto–tree accuracy.

- BLEU-4 (Papineni et al., 2002): The BLEU evaluation from e2e-metrics (Dušek et al., 2018).
- Tree accuracy (Balakrishnan et al., 2019): The ratio of annotated responses that pass the validity constraints specified by the input MR. Note that if constrained decoding terminates successfully, it is guaranteed to pass the tree accuracy check, but vanilla decoding comes with no such guarantee.
- Auto–tree accuracy: Tree accuracy after using a reverse model (trained on all the paired data) to parse the text. Note that parse errors make auto–tree accuracy less accurate than tree accuracy, but this method can be used with plain text output.

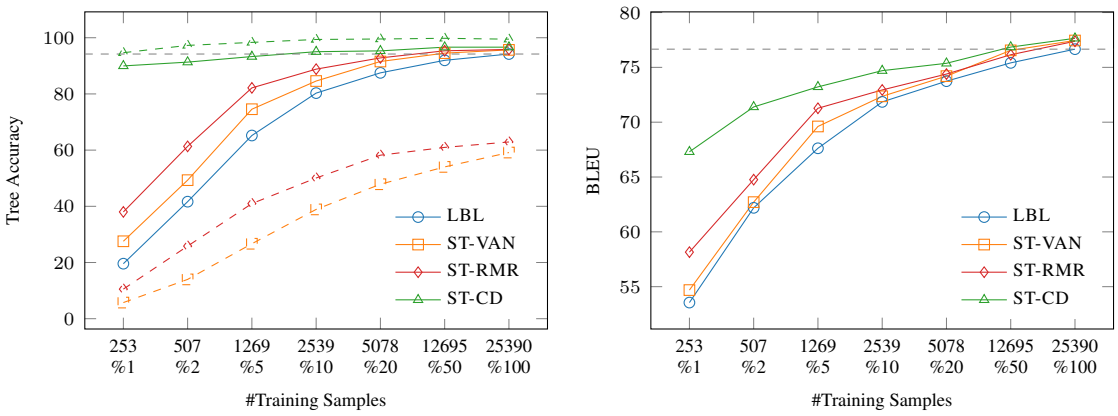


Figure 1: Tree accuracy and BLEU scores of the LSTM base model and three self-training strategies by parallel training data size with vanilla decoding on the conversational weather dataset. Tree accuracy on pseudo-labeled data is indicated by the same color dashed lines. Performance of the supervised model (LBL) using all of the labeled data is indicated by the gray dashed lines.

Implementation Our implementation[2] of self-training, constrained decoding and reverse model reranking is based on the same one-layer LSTM with attention approach as in Balakrishnan et al. (2019), with the same configuration of hyper-parameters. The experiments with pretrained models implement all above mentioned methods with BART (Lewis et al., 2020). We use the open source `fairseq` implementation (Ott et al., 2019). More specific configuration details of these two models are listed in Appendix A.

3.2 Data Efficiency Study

Figures 1 and 2 show the comparisons among the four training strategies on tree accuracy and BLEU score as a function of the amount of parallel data available. We can clearly see that ST-CD always surpasses the other three self-training methods. Meanwhile, the ST-CD lines are much flatter, indicating better data-efficiency, especially for tree accuracy with less parallel data. In particular, ST-CD achieves a considerable tree accuracy of 90% and 97% with LSTM and BART respectively, using only 1% of the parallel data (253 items). Using 100% of the data, ST-CD sets a new state-of-the-art in tree accuracy and BLEU, exceeding Rao et al.'s (2019) more complex tree-to-sequence method.[3]

Notably, with LSTM vanilla decoding, ST-CD needs only 20% of the parallel data to achieve comparable performance to LBL trained on all the parallel data.[4] More remarkably, BART vanilla decoding ST-CD needs only 2% of the parallel data to achieve essentially comparable performance to LBL trained on all the parallel data.[5] At this data efficiency level, tree accuracy exceeds 97% using just over 500 training samples, while Arun et al.'s (2020) results on the same dataset are under 90% despite using four times as much data. This is a key result since vanilla decoding is so much faster than constrained decoding, and latency is an important consideration for dialogue systems. For example, in our experiments using a single NVIDIA V100, the speed of LSTM vanilla decoding was 925.01 responses/s, or 37,973.22 tokens/s, while the speed of constrained decoding was 12.76 responses/s, or 532.61 tokens/s. This translates to an average of 80ms per response for constrained decoding, which is a barrier to production for systems with a strict latency budget. For BART, the speed of vanilla decoding was 25.17 responses/s, or 1565.75 tokens/s, while the speed of constrained decoding was 1.82 responses/s, or 113.92 tokens/s. As such, BART with vanilla decoding could be suitable in some production settings; alternatively, one could pursue knowledge distillation techniques as in Arun et al. (2020).

Although not as effective as ST-CD, ST-RMR also consistently surpasses ST-VAN and LBL. Moreover, it can also be used in more conventional

[2]Code is available at https://github.com/znculee/TreeNLG and https://github.com/znculee/TreeNLG-BART. See appendix for further details to enhance reproducibility.

[3]Results of using constrained decoding at runtime are shown in Figure 5 and Figure 6 in the appendix.

[4]With 20% of the parallel data, ST-CD exceeds LBL in tree accuracy while trailing it slightly in BLEU.

[5]Confirmed in significance tests on tree accuracy and human evaluation later in this section.

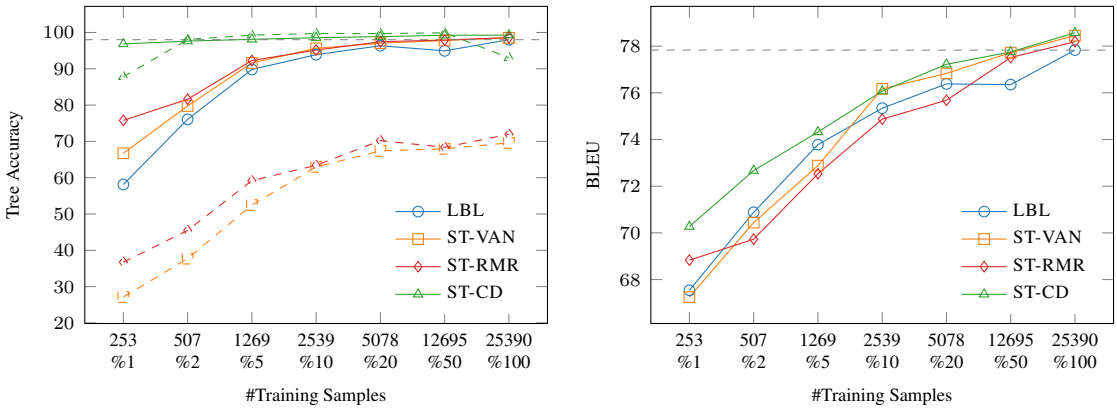


Figure 2: Tree accuracy and BLEU scores of the BART base model and three self-training strategies by parallel training data size with vanilla decoding on the conversational weather dataset. Tree accuracy on pseudo-labeled data is indicated by the same color dashed lines.

settings where the response text in the training data has no semantic annotations, and thus decoding is into plain text. As shown in Figure 3 (appendix), using auto–tree accuracy, ST-RMR can improve data efficiency when constrained decoding cannot be used. Note, however, that decoding into plain text consistently trails in auto–tree accuracy compared to decoding into annotated text.

3.3 How Does Self-Training Help?

Theoretically, self-training should be more helpful when the base model can produce higher quality pseudo-labeled data. As shown in Figures 1 and 2, tree accuracy on pseudo-labeled samples generated by ST-CD is much higher than other self-training methods, which illustrates why it yields much better tree accuracy and BLEU scores on the test set. Also note that the pseudo-labeled tree accuracy is much lower than the test tree accuracy for ST-VAN and ST-RMR. This may be because the unlabeled MRs are created by deletion and thus are somewhat atypical in comparison to the train and test sets.

3.4 Significance Tests

Although the gains in tree accuracy are large with vanilla decoding, to confirm that the gains in Figure 1 and 2 are significant, we have run McNemar's test (McNemar, 1947) comparing ST-CD against LBL as well as ST-VAN. Even when using LSTMs with 100% of the labeled data, the gain in tree accuracy from 94.2% with LBL to 96.6% with ST-CD is highly significant (p=4.30e-15), as is the gain from 95.7% with ST-VAN to 96.6% with ST-CD (p=0.0003). For BART, when using 100% of the labeled data, the gain in tree accuracy from 98.01% with LBL to 99.26% with ST-CD is highly significant (p=1.52e-7), as is the gain from 98.53% with ST-VAN to 99.26% with ST-CD (p=2.94e-4). Naturally, the gains when using less labeled data are also highly significant. Most interestingly, using only 2% of the labeled data with BART ST-CD is not significantly different than using 100% of the labeled data with BART LBL (p=0.68285).

3.5 Expert Evaluation of Correctness

In their experiments, Balakrishnan et al. (2019) found that tree accuracy differences reliably indicated differences in human evaluations of correctness, and in particular that tree accuracy failures nearly always indicated actual correctness errors. To verify these findings in our own targeted expert evaluation, we had two authors (both linguists) judge the correctness of the LSTM and BART models self-trained with constrained decoding using partial parallel data against the supervised baseline using the same partial parallel data and the best supervised model using all the parallel data, where the judges were blind to which model was which. Correctness was judged against the reference text for 50 randomly selected pairs in each condition where the items differed in tree accuracy. For each pair, the judges indicated whether the first item was better than, the same as or worse than the second item. 3-way agreement was 79% for correctness between the judges; moreover, when excluding any 'same' judgments, the judges agreed in all but one case. After the judgments were collected, we calculated how well they agreed with tree accuracy, excluding the indeterminate 'same' judgments. Agreement was quite high, reaching

90% for one judge and 88% for the other. (Further details are in Appendix B.) Given this high level of agreement with the automatic tree accuracy measure along with the highly significant differences in tree accuracy, we focused our human evaluation on investigating whether the observed differences in BLEU scores indicated important differences in grammaticality.

3.6 Human Evaluation of Grammaticality

While the BART ST-CD-02 and LSTM ST-CD-20 models achieved comparable or better levels of tree accuracy in comparison to their LBL-100 (full-data) counterparts, they trailed somewhat in BLEU scores. Looking at the outputs of the self-trained models with the worst BLEU scores, we found that the responses were mostly good, only suffering from clear grammaticality issues infrequently. To confirm these observations, we conducted a human evaluation using the responses generated by the BART ST-CD-02 and LSTM ST-CD-20 models on 333 randomly selected test items, along with the responses for the same items for the best and worst supervised models by BLEU score, namely BART LBL-100 and LSTM LBL-01. Mixed in with the responses of each model were 75 check items, 25 of which were grammatical and 50 of which we intentionally made ungrammatical.

Using these samples, we ran an experiment on Amazon Mechanical Turk involving 16 unique participants. The participants in the experiment were pre-filtered by selecting those with an approval rate of at least 95%. Each participant was shown our grammaticality guidelines, which were based on Arun et al.'s (2020) and available for review at all times during the experiment. They were subsequently asked to take a quiz. Those who scored 80% or more on the quiz were selected for further participation. To encourage careful engagement with the task, we offered bonus payments to those who performed well on the check items. The experiments were carried out with Institutional Review Board approval, and all participants were paid above minimum wage for our locale.

Agreement with the check items was quite robust, with all participants well above chance, though there were some outliers with respect to check item agreement. This indicates that the judgments were somewhat noisy. Each item received 3 judgments, and the items were assigned the majority judgment for analysis purposes. Judgments of ungrammaticality were accompanied by brief reasons; discrepancies between judgments primarily reflected difficulty in applying the guidelines regarding punctuation.

Our results indicate that 4.8% of the BART ST-CD-02 items were judged ungrammatical, not far from the error rates of 3.9% for LSTM ST-CD-20 and 3.0% for BART LBL-100. By contrast, 11.4% of the LSTM LBL-01 items were judged ungrammatical. Pairwise comparisons using McNemar's test only revealed statistically significant differences for the LSTM LBL-01 model: it was judged significantly worse than the 3 other models ($p < 0.003$ in all cases), while none of the other systems were significantly different ($p > 0.3$ in all cases).

3.7 Qualitative Analysis

The most frequent grammaticality issue, especially for LSTM ST-CD-20, was missing punctuation between independent clauses, as shown in (a) in Table 2. Other errors included occasional agreement errors or missing constituents, as in (b). Example correctness errors appear in Table 3; they usually involved missing information, but sometimes also repeated or extra information.

3.8 E2E Confirmation

We also evaluate our strategies on the enhanced E2E dataset. As shown in Figure 4 in Appendix, we can draw the same general conclusions regarding data efficiency as with the conversational weather dataset.[6] Both constrained decoding and reverse model reranking improve upon vanilla self-training, with constrained decoding being more effective when using less parallel data. Notably, for LSTM models, with vanilla decoding at runtime, tree accuracy and BLEU of using self-training with constrained decoding and 20% of the parallel data (ST-CD-20) are essentially identical to the supervised model using all the available data (LBL-100). For BART models, the performance of ST-CD-02 is also very similar to the one of LBL-100: While the BLEU score of ST-CD-02 is slightly lower than that of LBL-100, it is still very high, and the tree accuracy of ST-CD-02 is slightly higher than the tree accuracy of LBL-100.

[6]Note that the BLEU scores here are calculated in the same generous way as in Balakrishnan et al.'s (2019) evaluation. In particular, since multiple test MRs in the enhanced data have the same original MR, we select the best generation of the same original MR using NLTK's (Bird et al., 2009) implementation of sentence BLEU on multi-references.

Index	System	Error	Reason
(a)	LSTM ST-CD-20	No , the forecast does not call for sunny skies expect partly cloudy skies	Punctuation is missing before *expect*.
(b)	BART ST-CD-02	Today in ARG_CITY will have a high of ARG_TEMP_HIGH and a low of ARG_TEMP_LOW	Missing subject.

Table 2: Examples of grammaticality errors

Index	System	Error	Reference
(a)	LSTM LBL-20	Yes , it will be mostly sunny today in your area	Yes , it will be mostly sunny today and ARG_WEEKDAY in your area
(b)	LSTM LBL-100	Yes , light rain is likely today , and light thunderstorms and rain are likely on ARG_WEEKDAY and light thunderstorms and rain are likely on ARG_WEEKDAY	Yes , light rain is likely today . ARG_WEEKDAY will also have light rain and light thunderstorms and rain are likely on ARG_WEEKDAY

Table 3: Examples of correctness errors

4 Related Work

There is a much more established tradition of using self-training in parsing, where McClosky et al. (2006) and subsequently others have shown that that self-training can yield substantially improved parsing accuracy. In NLG, Kedzie and McKeown (2019) and Qader et al. (2019) pursue self-training for data efficiency but only using flat input representations and without constrained decoding, as noted earlier. Qader et al. (2019) develop a sophisticated, joint method of self-training NLG and NLU models. Kedzie and McKeown (2019) make use of noise injection sampling and NLU models to create new MR-text pairs, where the new MRs often contain fewer slots than the original MR; here, we similarly create new, simpler MRs, but do so directly by just deleting nodes in the input trees. Likewise, our general approach to self-training (He et al., 2020) is much simpler than in Chang et al.'s (2021) work, where they generate new text samples using GPT-2 (unconditioned on any input) then pair them with data samples. Earlier, Chisholm et al. (2017) train NLG and NLU models that share parameters to reduce the risk of hallucination. Our iterative method of training forward and reverse seq2seq models instead draws from Yee et al.'s (2019) reverse model reranking approach and is much simpler to implement. Additionally, Nie et al. (2019) apply self-training to a NLU model to reduce the noise in the original MR-text pairs in order to reduce the hallucination problem in NLG, but they do not investigate data efficiency issues. Also related is work on back-translation (Sennrich et al., 2016) in MT, which starts from the assumption that there is much target side data; by contrast, self-training assumes there is much source side data, as is the case with our task (where new unlabeled MRs can be easily created).

More recent work takes advantage of pre-trained language models to develop few-shot NLG methods. Chen et al. (2019) show impressive results with just 200 training items using a specialized table encoder with GPT-2, while Peng et al. (2020) use cross-domain training (an orthogonal approach) together with GPT-2; neither investigates more challenging compositional inputs. Although Arun et al. (2020) do use BART on compositional inputs, their tree accuracy levels are much lower even when using considerably more data.

More generally, reverse (or reconstructor) models have taken on greater theoretical interest thanks to Rational Speech Act (RSA) theory (Frank et al., 2009) and have recently proved useful in NLG (Fried et al., 2018; Shen et al., 2019). Our approach differs in using reverse models during self-training rather than at runtime. Work on combining parsing and generation for ambiguity avoidance goes back much farther (Neumann and van Noord, 1992), with managing the trade-off between fluency and ambiguity avoidance a more recent topic (Duan and White, 2014) that we also leave for future work. Constrained decoding (Balakrishnan et al., 2019) is inspired by coverage tracking in grammar-based approaches to realization (Kay, 1996; Carroll and Oepen, 2005; White, 2006); its use during self-training is novel to this work.

5 Conclusion and Future Work

In this paper, we have shown that using self-training with constrained decoding in compositional neural NLG can deliver large gains in data efficiency, enabling seq2seq models to achieve satisfactory quality using vanilla decoding with much less annotated data. The idea of using constrained decoding with self-training rather than for runtime inference is a very simple one, but ours is the first paper to investigate the idea, and we show via thorough experimentation and evaluation that it works remarkably well. In our experiments, we found that LSTM models trained from scratch can increase data efficiency by a factor of at least 5, while pre-trained BART models yielded a 50 times increase, achieving essentially comparable levels of correctness and grammaticality using only 2% of the existing training data. As such, the approach promises to help pave the way towards developing systems with mere hundreds rather than tens of thousands of annotated samples, potentially eliminating the need for crowdsourcing in system development. In future work, it would be exploring ways of at least partially automatically adding semantic annotations to the target texts using methods that treat such annotations as latent (Shen et al., 2020; Xu et al., 2021) to facilitate using our approach on a new task or dataset.

Acknowledgements

We thank that the Ohio Super Computer Center (Center, 1987) supports us sufficient computational devices for training many large models in our experiments. This research was supported by a collaborative open science research agreement between Facebook and The Ohio State University. The last author is a paid consultant for Facebook.

References

Ankit Arun, Soumya Batra, Vikas Bhardwaj, Ashwini Challa, Pinar Donmez, Peyman Heidari, Hakan Inan, Shashank Jain, Anuj Kumar, Shawn Mei, Karthik Mohan, and Michael White. 2020. Best practices for data-efficient modeling in NLG:how to train production-ready neural models with less data. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 64–77, Online. International Committee on Computational Linguistics.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844, Florence, Italy. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Giuseppe Carenini and Johanna D. Moore. 2006. Generating and evaluating evaluative arguments. *Artificial Intelligence*, 170:925–952.

John Carroll and Stefan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proc. IJCNLP-05*.

Ohio Supercomputer Center. 1987. Ohio supercomputer center.

Ernie Chang, Xiaoyu Shen, Dawei Zhu, Vera Demberg, and Hui Su. 2021. Neural data-to-text generation with LM-based text augmentation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 758–768, Online. Association for Computational Linguistics.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2019. Few-shot nlg with pre-trained language model.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from Wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 633–642, Valencia, Spain. Association for Computational Linguistics.

Vera Demberg, Andi Winterboer, and Johanna D Moore. 2011. A strategy for information presentation in spoken dialog systems. *Computational Linguistics*, 37(3):489–539.

Manjuan Duan and Michael White. 2014. That's not what I meant! Using parsers to avoid structural ambiguities in generated text. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 413–423, Baltimore, Maryland. Association for Computational Linguistics.

Ondřej Dušek and Filip Jurcicek. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG Challenge. In *Proc. of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg,

The Netherlands. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2019. Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG Challenge. *arXiv preprint arXiv:1901.11528*.

Michael Frank, Noah Goodman, Peter Lai, and Joshua Tenenbaum. 2009. Informative communication in word production and word learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, pages 1228–1233.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018. Unified pragmatic models for generating and following instructions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1951–1963, New Orleans, Louisiana. Association for Computational Linguistics.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *International Conference on Learning Representations*.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 200–204, Santa Cruz, California, USA. Association for Computational Linguistics.

Chris Kedzie and Kathleen McKeown. 2019. A good sample is hard to find: Noise injection sampling and self-training for neural language generation models. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 584–593, Tokyo, Japan. Association for Computational Linguistics.

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Hongyuan Mei, Mohit Bansal, and R. Matthew Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730. Association for Computational Linguistics.

Gunter Neumann and Gertjan van Noord. 1992. Self-monitoring with reversible grammars. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*, pages 700–706.

Feng Nie, Jin-Ge Yao, Jinpeng Wang, Rong Pan, and Chin-Yew Lin. 2019. A simple recipe towards reducing hallucination in neural surface realisation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2673–2679, Florence, Italy. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog.

Raheel Qader, François Portet, and Cyril Labbé. 2019. Semi-supervised neural text generation by joint learning of natural language generation and natural language understanding models. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 552–562, Tokyo, Japan. Association for Computational Linguistics.

Jinfeng Rao, Kartikeya Upasani, Anusha Balakrishnan, Michael White, Anuj Kumar, and Rajen Subba. 2019. A tree-to-sequence model for neural NLG in task-oriented dialog. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 95–100, Tokyo, Japan. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4060–4067, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. 2020. Neural data-to-text generation via jointly learning the segmentation and correspondence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7155–7165, Online. Association for Computational Linguistics.

Marilyn Walker, Amanda Stent, Francois Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research (JAIR)*, 30:413–456.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, M. Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 120–129. Association for Computational Linguistics.

Michael White. 2006. Efficient realization of coordinate structures in combinatory categorial grammar. *Research on Language and Computation*, 4(1):39–75.

Michael White, Robert A. J. Clark, and Johanna D. Moore. 2010. Generating tailored, comparative descriptions with contextually appropriate intonation. *Computational Linguistics*, 36(2):159–201.

Xinnuo Xu, Ondřej Dušek, Verena Rieser, and Ioannis Konstas. 2021. AggGen: Ordering and aggregating while generating. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1419–1434, Online. Association for Computational Linguistics.

Kyra Yee, Yann Dauphin, and Michael Auli. 2019. Simple and effective noisy channel modeling for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5696–5701, Hong Kong, China. Association for Computational Linguistics.

A Reproducibility Details

For LSTM models, the word embedding and hidden size dimensions are 300 and 128 respectively, and the decoder output embedding size is 512. The dropout rate for both encoder and decoder is 0.2. There are no more than 128 sentences in a batch. Training uses early stopping when the validation loss has not improved for the last 20 epochs. The learning rate is 0.001, and the scheduler is *ReduceLROnPlateau* whose factor is 0.1 and patience is 3. The maximum output length is 2 times source length plus 50, and the beam size is 5. The loss function is optimized with Adam (Kingma and Ba, 2014), where $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

For BART models, we use the BART-Large model available in the fairseq, which 12 encoder and decoder layers.[7] The dropout rate for both encoder and decoder is 0.1. There are no more than 2048 tokens in a batch. Training uses early stopping when the validation loss has not improved for the last 20 epochs. The learning rate is 0.00003, and the scheduler is *polynomial decay* with 1000 warm updates. The maximum output length is 1024. The loss function is optimized with Adam (Kingma and Ba, 2014), where $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

For every experiment, the computing infrastructure we used is an NVIDIA V100 GPU and an Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz CPU. The numbers of trainable parameters of LSTM models for weather and E2E datasets are 2,212,928 and 3,079,256 respectively. Training a LSTM model on the full weather dataset takes around 0.5k seconds for 38 epochs. Training a LSTM model on the pseudo-labeled weather dataset takes around 3.4k seconds for 57 epochs. Training and validation loss at convergence is around 1.8. The speed of vanilla decoding was 37,973 tokens/s, and the speed of constrained decoding was 532.61 tokens/s. The numbers of trainable parameters of BART models for weather and E2E datasets are both 406,290,432. Training a BART model on the full weather dataset takes around 10k seconds for 21 epochs. Training a BART model on the pseudo-labeled weather dataset takes around 42k seconds for 20 epochs. Training and validation loss at convergence is around 2.1. The speed of vanilla decoding was 25.17 responses/s, or 1565.75 tokens/s, and the speed of constrained decoding was 1.82 responses/s, or 113.92 tokens/s.

The source code and data for reproducing all experiments in this paper are submitted in the supplementary materials and will

[7] https://github.com/pytorch/fairseq/tree/master/examples/bart

be released upon acceptance. The dependencies are specified in `requirements.txt`. Code usage instructions are in `README.md` and `self-training/README.md`.

B Details on Expert Evaluation of Correctness

Table 4 shows the detailed breakdown of agreement between the expert judges and tree accuracy. We can observe that agreement with tree accuracy is higher with LSTM models than with BART, and higher where there is a significant difference in tree accuracy than in the one case where there was no significant difference (BART ST-CD-02 vs. BART LBL-100). For this comparison, there were relatively few discrepancies in tree accuracy to sample from, and the items in question likely represent somewhat unusual cases. In examining the handful of cases where the judges agreed but did not agree with tree accuracy, about half were real errors where BART's words did not match the non-terminals (influenced by its pre-trained knowledge), while the other half had (presumably rare) errors in the input or reference. It is not surprising that tree accuracy would be somewhat less reliable with BART, as it relies on its pre-trained knowledge as well as the input in making generation choices. For example, in one case the BART ST-CD-02 model output, "It's not expected to be warm tomorrow morning in your area. The temperature will drop to __ARG_TEMP__ tomorrow." Here, it seems that BART inferred that if it won't be warm tomorrow, that may well be because the temperature is dropping. However, "will drop" is not part of the input and may or may not be factual. Since these words appear outside of the non-terminal signaling the low temperature in the output, they are not checked by tree accuracy, and thus this error is missed.

LSTM	ST-CD-20 vs.	
	LBL-20	LBL-100
Judge-1	36/39	26/29
Judge-2	40/40	25/27

BART	ST-CD-02 vs.	
	LBL-02	LBL-100
Judge-1	36/37	21/31
Judge-2	36/37	18/29

Table 4: Agreement rate of human evaluation of correctness with tree accuracy (excluding indeterminate 'same' judgments)

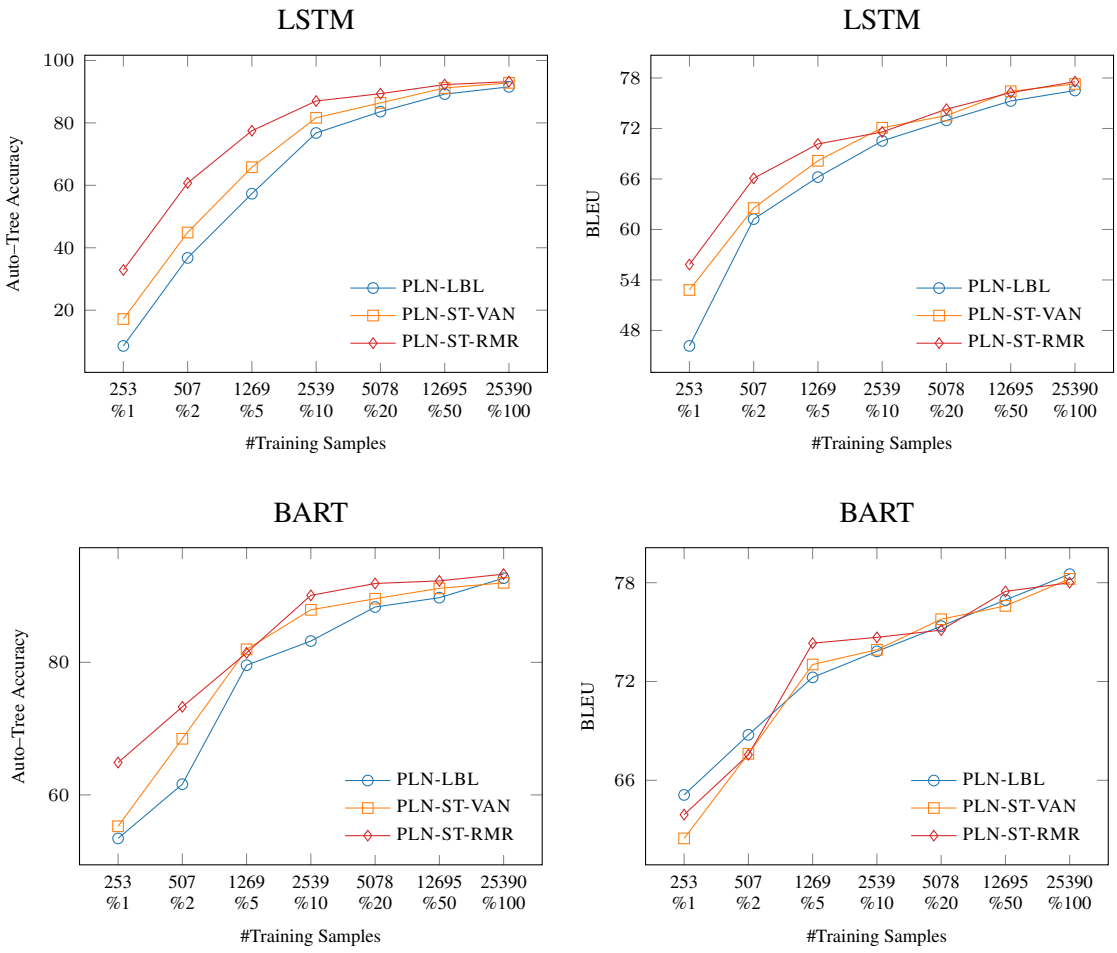


Figure 3: Auto–tree accuracy and BLEU scores of LSTM and BART models and two self-training methods by parallel training data size with vanilla decoding of plain (PLN) text on the conversational weather dataset. For comparison, auto–tree accuracy of LSTM and BART on the test set references are 92.85 and 93.18 respectively.

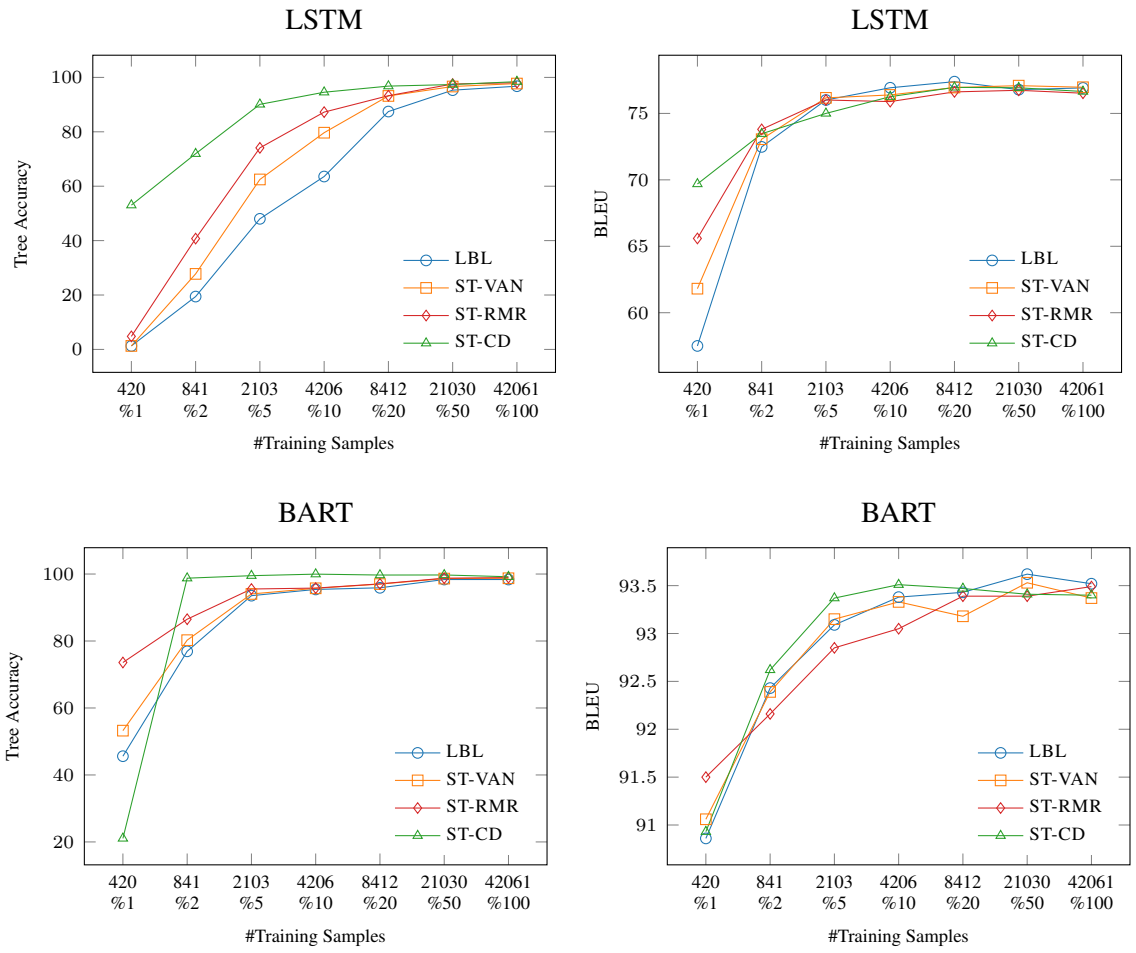


Figure 4: Tree accuracy and BLEU scores of LSTM and BART models and two self-training strategies by parallel training data size with vanilla decoding on the enhanced E2E dataset. The self-training results here are measured on the first iteration.

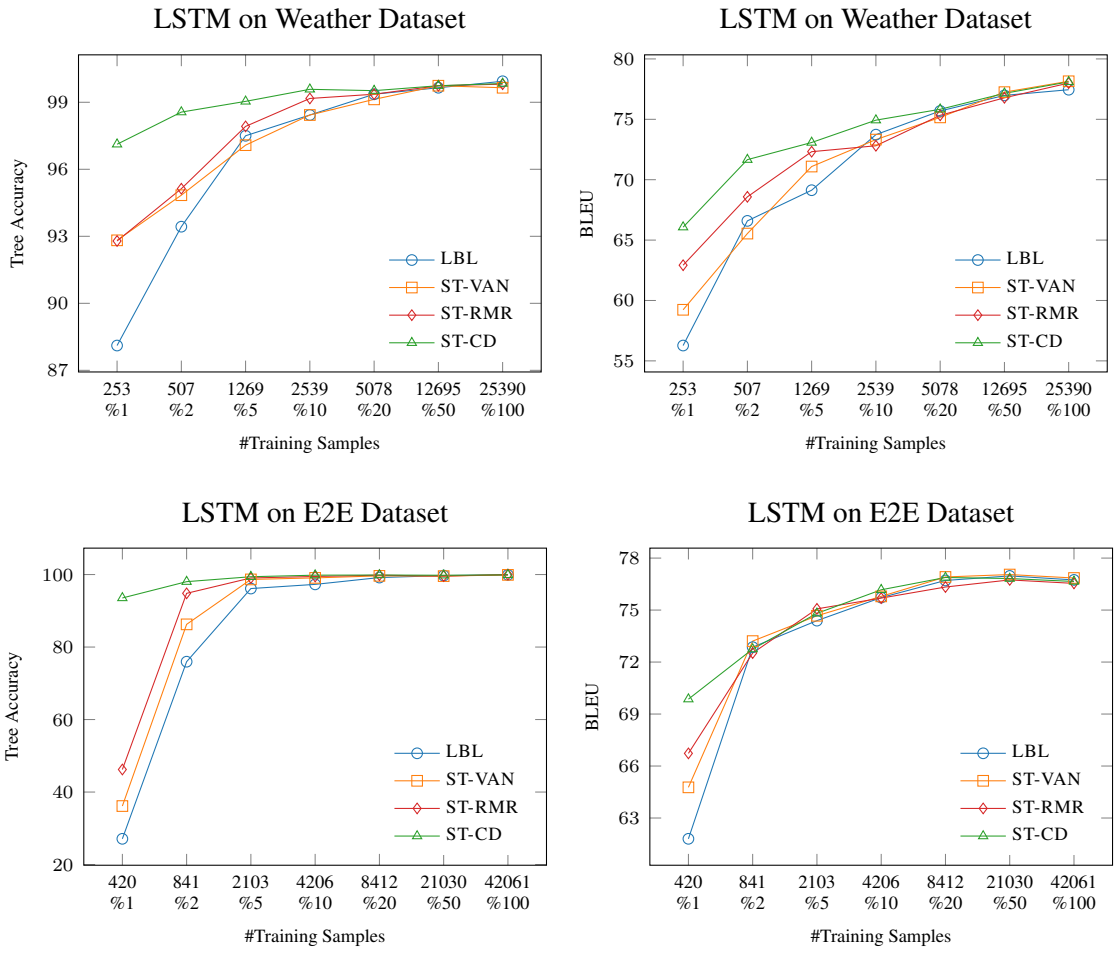


Figure 5: Tree accuracy and BLEU scores of LSTM and two self-training strategies by parallel training data size with **constrained decoding** at runtime on the conversational weather dataset and the enhanced E2E dataset. The self-training results of the enhanced E2E dataset are measured on the first iteration.

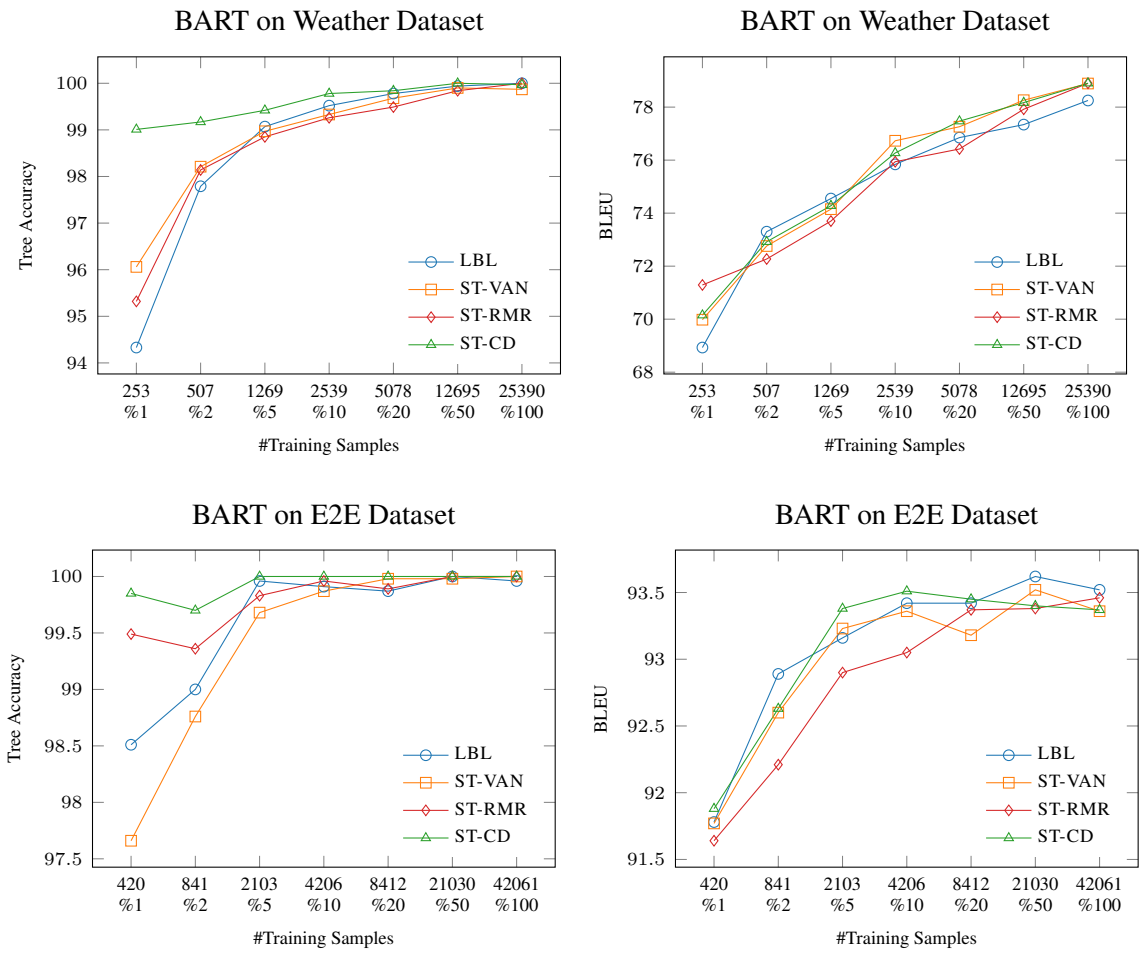


Figure 6: Tree accuracy and BLEU scores of BART and two self-training strategies by parallel training data size with **constrained decoding** at runtime on the conversational weather dataset and the enhanced E2E dataset. The self-training results of the enhanced E2E dataset are measured on the first iteration.

Generating Racing Game Commentary from Vision, Language, and Structured Data

Tatsuya Ishigaki[†] **Goran Topić**[†] **Yumi Hamazono**[†◦] **Hiroshi Noji**[†◇]
Ichiro Kobayashi[†◦] **Yusuke Miyao**[†‡] **Hiroya Takamura**[†]

[†]National Institute of Advanced Industrial Science and Technology, Japan,
[◦]Ochanomizu University, [‡]The University of Tokyo, [◇]LeepMind Inc.
{ishigaki.tatsuya, goran.topic, hamazono-yumi, noji, takamura.hiroya}@aist.go.jp,
koba@is.ocha.ac.jp, yusuke@is.s.u-tokyo.ac.jp

Abstract

We propose the task of automatically generating commentaries for races in a motor racing game, from vision, structured numerical, and textual data. Commentaries provide information to support spectators in understanding events in races. Commentary generation models need to interpret the race situation and generate the correct content at the right moment. We divide the task into two subtasks: utterance timing identification and utterance generation. Because existing datasets do not have such alignments of data in multiple modalities, this setting has not been explored in depth. In this study, we introduce a new large-scale dataset that contains aligned video data, structured numerical data, and transcribed commentaries that consist of 129,226 utterances in 1,389 races in a game. Our analysis reveals that the characteristics of commentaries change depending on time and viewpoints. Our experiments on the subtasks show that it is still challenging for a state-of-the-art vision encoder to capture useful information from videos to generate accurate commentaries. We make the dataset and baseline implementation publicly available for further research.[1]

1 Introduction

Live commentary plays an important role in sports matches and video games; it makes spectators more excited, more immersed, and better informed about the matches or games (e.g., Schaffrath (2003)), as in the example of racing game commentary "*We are approaching the final long straight. I wonder who is going to win!*". Live commentary also enhances the value of online videos and home videos. However, providing a live commentary requires a certain level of commenting skills and knowledge of the target sports or video games; the majority of online videos and home videos are left without live commentary.[2] The application of natural language generation technology would be a solution to this problem. Thus, we select the racing game domain as an example, and propose a task of generating live commentary on it.

Examples of utterances in a live commentary for a racing game are shown in Figure 1. Live commentaries should describe each important event in the race at the moment when the event occurs, within a short period of time. Thus, we have to make decisions on *when to speak* and *how long/elaborately to speak*, in addition to *what to say* and *how to say it*, which have been long studied. The importance of each event would have to be assessed in the context of a competition in which participants are striving to win. It suggests that *what to say* for race commentary generation should be different from, for example, image captioning. In this sense, the task of live commentary generation contains inherent limitations that are not addressed in well-studied generation problems such as summary generation for basketball (Puduppully and Lapata, 2021), and image/video captioning (Vinyals et al., 2015; Yao et al., 2015); however, some techniques developed for such existing generation tasks are also useful for live commentary generation.

As input, vision data, such as the videos shown in Figure 1, are common in many tasks. However, it is not a trivial task to capture important information from vision data, because many of the frames in a race would be similar to each other, unlike images for image captioning data. Therefore, we propose the use of structured data, that is, telemetry data, including the positions and the speeds of cars, and the steering wheel angles. The

[1] https://kirt.airc.aist.go.jp/RacingCommentary

[2] For example, many gameplay videos on Twitch do not have live commentary (https://www.twitch.tv).

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 103–113,
Aberdeen, Scotland, UK, 20-24 September 2021.

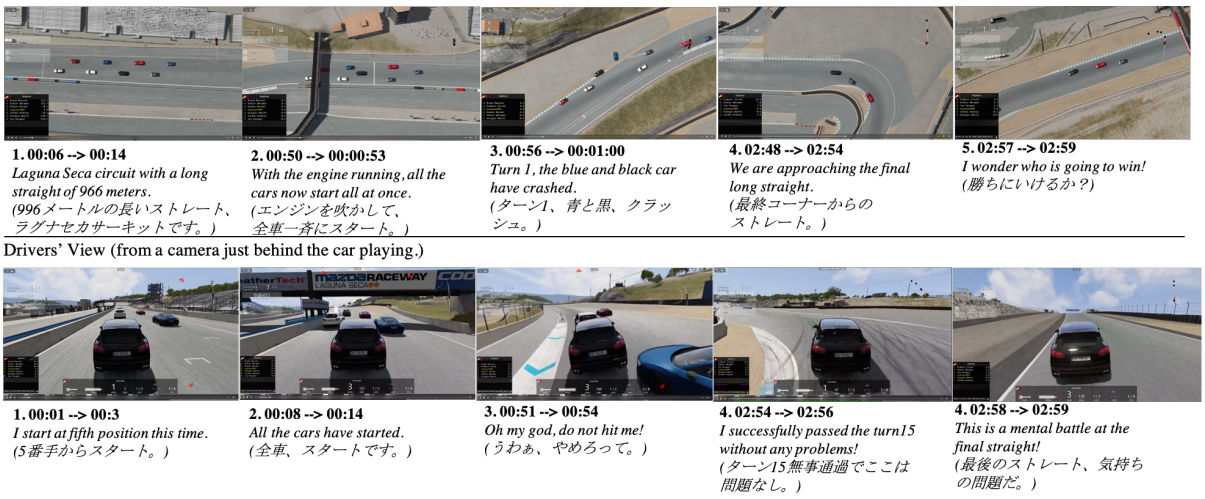


Figure 1: Translated Examples of commentaries (original utterances in Japanese are in brackets). For the commentaries on the top, the commentator is watching the race from the aerial viewpoint. For those at the bottom, the commentator is watching the race through a camera just behind the driver.

assumption that such telemetry data are available is not unrealistic. It is the general trend that many sensors are used to obtain telemetry data even in real sports matches and motor races. For example, each race car in F1 races is monitored by 300 sensors.[3] Additionally, players are tracked by GPS technology during football matches to obtain positional data (Memmert and Raabe, 2018). We work on video games because telemetry or vision data are easier to obtain than in real sports. This can address the huge demand in the gaming community and serve as a favorable test bed for live commentary generation. As a result, the task addressed in this paper is the live commentary generation for racing games from vision, structured, and textual data.[4]

Live commentary generation has not been studied in-depth, partly because of the lack of datasets. Thus, we create a new dataset for live commentary generation, which includes 129,226 utterances of live commentary, aligned with gameplay video and telemetry data of racing game. The telemetry data contain the positions and speeds of race cars and various types of information about circuits and cars. There are two types of live commentary. One is provided by the game players while playing and watching the racing game from the virtual camera behind the car. The other is provided by another person watching the game from a virtual helicopter. We analyze the differences in the characteristics of commentaries from different viewpoints.

We split the live commentary generation into two subtasks: the utterance timing identification and utterance generation. We propose multimodal models for these subtasks and also provide an empirical evaluation. Our experiments suggest that the use of telemetry data works well for this task, whereas it is difficult for a state-of-the-art vision encoder to extract useful information from race videos, especially for utterance generation.

Our contributions are threefold: (1) we propose a novel task of automatically generating motor racing game commentaries, (2) we create a dataset and analyze its characteristics, and (3) we propose methods for this task and argue that combining multimodal data is challenging, which is worth exploring in depth. We make the dataset and baseline implementations publicly available to enhance further studies on this task.

2 Related Work

Existing studies on commentary generation can be divided into real-time commentary (Taniguchi et al., 2019; Kim and Choi, 2020) and commentaries written afterwards (Puduppully and Lapata, 2021). Our focus is on the former. Live commentary generation is formulated as the extraction of tweets (Kubo et al., 2013), the combination of rules and keyword extraction from videos (Kim and Choi, 2020) and neural network-based data-to-text (Taniguchi et al., 2019). To

[3] https://aws.amazon.com/f1/

[4] We include textual data as input because we use the previous utterances as additional input.

generate commentary in real-time, we need to solve at least two tasks: timing identification and utterance generation tasks. However, existing studies focus on the latter, where the timings are given, for example, minute-by-minute updates (Kubo et al., 2013). Unlike baseball, the timing identification task for race commentary is not trivial because a race cannot be segmented simply.

Our setting can be considered as a combination of two different research topics: video captioning (Kim and Choi, 2020) and data-to-text (Taniguchi et al., 2019). Various methods for encoding video frames have been actively studied (Dosovitskiy et al., 2021); commentaries often include comments that focus on the positional relation between cars, which requires a more fine-grained understanding of video frames. The performance of current vision encoders still needs to be evaluated. Data-to-text is the task of converting structured data into natural language, which has been applied to the domain of finance (Murakami et al., 2017; Aoki et al., 2018, 2021; Uehara et al., 2020), weather forecast (Murakami et al., 2021), a summary of sports matches (Puduppully and Lapata, 2021; Iso et al., 2019) and live sports commentary (Taniguchi et al., 2019). The inputs used for existing studies are time-sequence numerical data (Murakami et al., 2017), tables (Puduppully and Lapata, 2021; Gardent et al., 2017) or simulated images (Murakami et al., 2021). These models focus on neural network-based approaches; however, data-to-text tasks have been studied for a long time (see a survey paper (Gatt and Krahmer, 2018) for details).

Existing studies on generation mostly focus on generating text from a single viewpoint, i.e. they generate objective descriptions of video frames in video captioning, and a data-to-text model generates a text that focuses on the main content of the input data. A few existing studies state that live commentaries change depending on the viewpoints of commentators. For example, Kubo et al. (2013) found that the generated commentaries on soccer matches are not objective, and these are biased to mention more popular teams. The viewpoints are the key to characteristic commentaries, but most studies have ignored the difference caused by the viewpoints that our dataset addresses.

Datasets play important roles in studies on generation. Existing datasets for generation tasks contain data in a single modality, such as, videos (Zhou et al., 2018; Krishna et al.) or structured data (Puduppully and Lapata, 2021; Gardent et al., 2017). We propose a new large-scale dataset that contains transcribed commentaries aligned with videos and structured numerical data.

3 Dataset

We describe the procedure used to create our dataset. We then show its statistics and the analysis to characterize the task.

3.1 Procedure for creating our dataset

Collecting recordings and spoken commentaries: We hired five workers who regularly play e-sports games. Thus, some of the workers are familiar with playing racing games, but some are not. They are not professional commentators. As a racing game, we used Assetto Corsa[5]. For each race consisting of two laps, one worker plays while simultaneously commentating it from the viewpoint of the virtual camera just behind the car (*driver's view*). Another worker is assigned to commentate the race from the viewpoint of a virtual helicopter (*aerial view*), without playing the game. Note that the commentaries are in Japanese. Drivers used a physical steering controller to achieve a situation close to real sports competitions.

For both commentaries, we ask the commentators to mainly mention the car driven by the player; however, the commentators could also mention other cars if they found them worth mentioning. Circuit maps, in which each turn is numbered, are available to commentators so that they can refer to them by numbers (e.g., *Turn 15*). Well-known turns or straights are given names such as *Casanova* for turn six in the Laguna Seca circuit.[6]

Collecting transcriptions of commentaries: After the collection of recordings, we hired 149 workers on a crowdsourcing service, Lancers[7], to transcribe all the recordings. Workers are supposed to transcribe the recordings and add the

[5] Assetto Corsa is a game title developed and published by Kunos Simulazioni:
http://www.kunos-simulazioni.com

[6] The numbers and names are obtained from Wikipedia or other websites that describe circuits.

[7] http://lancers.jp

Telemetry data types	Example values
current lap [0..]	1
is current lap invalidated?	false
lap time of current lap (ms)	256
lap time of previous lap (ms)	156164
progress on current lap [0, 1]	0.002780
projected diff. from best lap	0.0
speed (km/h)	177.693130
steer rotation (rad)	-59.793526
world position (x, y, z) (m)	(5.372770, 64.056038, -749.219971)
position on track (L=−1, R=1)	-0.515301
distance from ideal path (m)	0.854022

Table 1: List of collected structured telemetry data with example values. The last two types of data are not from the API, but are calculated by the authors.

# of unique circuits	4
# of commentators	5
total # of races	1,389
total # of recordings	2,473
- driver's viewpoint	1,389
- aerial viewpoint	1,084
total recording duration	226:37:53
- driver's viewpoint	126:11:19
- aerial viewpoint	100:26:34
total # of utterances	129,226
avg. # of utterances per race	52.25
avg. # of characters per utterance	22.22
avg. length of an utterance	2.73s
avg. length of silence	3.46s

Table 2: Statistics of the dataset.

start and end timestamps to each utterance. Utterances are basically sentences, with some exceptions; some utterances do not form complete sentences because they are truncated owing to speech repair. Finally, we manually checked whether the transcriptions aligned with the videos correctly.

Collecting structured telemetry data: We also collected structured telemetry data. Using Assetto Corsa's API, we extracted various structured numerical data, including the speeds of the cars participating in the race, % of the progress over the entire race, the angles of the steering wheel, and other 13 types of numerical values. The full list of the types of structured data collected is shown in Table 1. We repeated the extraction of these values every 0.01 seconds on average.

3.2 Statistics and Analysis

In total, the five workers had played 1,389 races. 1,084 out of the 1,389 races are given commentaries from both the drivers' and aerial viewpoints. The remaining 305 races are given only commentaries from the drivers' viewpoints. Thus, we collected a total of 2,473 video recordings aligned with commentaries and multimodal data.[8] The total duration of the recordings is 226 hours, and the number of collected utterances is 129,226, which is more than the number of descriptions in ActivityNet Captions dataset (Krishna et al.), a large dataset for dense video captioning. Also, as a non-English dataset, it might provide some valuable linguistic diversity, as most datasets are in English. On average, they produced an utterance with a length of 2.73 seconds and then they kept silent for 3.46 seconds. The other statistics are listed in Table 2.

We manually designed labels for the utterances to capture their characteristics. Each label is defined as a pair of two sub-labels, target label and the content label, as presented in Table 3.

The target label represents the target subject of the utterance, such as the `player's car`, `other cars`, `all cars`, or `the circuit`. For example, the utterance *"All the cars now start"* is labeled as `all cars`, because it focuses on all the cars participating in the race, whereas *"The player is now approaching Turn15"* is labeled as the `player's car`, because it mentions only the target car. The content label represents the content of the utterance, such as the `relative position`, `movement`, `lap time` and other content types as presented in Table 3. As an example, *the player is now approaching Turn15* is labeled as the *player's car:movement*, because it mentions the movement of the player's car.

We randomly extracted 874 utterances from 20 videos, and then manually annotated them using the listed labels. It should be noted that this manual labelling task is performed under the multi-label setting, which allows us to assign one or more labels to an utterance. We analyze the distributions of the labels to capture the characteristics of the dataset. In this analysis, we are particularly interested in (1) how the label distribution changes over time, and (2) how the label distribution differs depending on the commentator's viewpoint.

How utterances change over time?

We split a race into quarters according to the timeline (e.g., the first quarter corresponds to the interval from the beginning of a race to the 25% point).

[8] 1,084+305 videos from driver's viewpoints, and 1,084 videos from aerial viewpoints.

Target labels	Example utterances
player's car	*This was a very elegant overtake by the player.*
other cars	*The car behind just overtook the player.*
all cars	*All the cars has now started.*
circuit	*Laguna Seca is well known for its very long strait.*
none	*Ah!*
Content labels	**Example utterances**
relative position	*The player is now at the second making the distance close to the first.*
location on map	*The blue car is approaching Turn2 and others follow.*
lap time	*The player now crossed the finish line at the time 3.15*
previous event	*Maybe this mistake might cause big impact on the time*
future event	*Can the player successfully pass the difficult Turn 15?*
movement	*The player overtook the red* car on this long straight.
stable race	*All cars are stable without any problems.*
features	*All the cars are the same, Porsche Macan.*
greetings	*Ok, now I start my commentary on this race.*
reaction	*Oh!*
others	—

Table 3: List of sub-labels and example utterances. A label assigned to an utterance is defined as a pair of Target and Content sub-labels.

Figure 2 shows the label distributions for different quarters. In the figure, the proportions of the labels in the first quarter are represented by the tops of the four bars, which are colored blue. Similarly, the second (orange), third (black), and fourth (yellow) bars from the top represent the proportions in the second, third, and fourth quarters, respectively.

For the first quarter indicated by the top bar for each label, which are colored blue in the figure, the labels with *features* (i.e., *circuit:features*, *player's car:features* and *other cars:features*) are frequent compared with the other quarters. This suggests that commentators often start the commentary by mentioning the features of the circuit or the cars.

For the final quarter indicated by the bottom bars, which are colored yellow, *none:greetings* and *player's car:lap time* are frequent, suggesting that the commentators mention the elapsed time after the cars crossed the finish line and finally concluded the session with greetings.

Next, we focus on the differences between the two middle quarters, indicated by the second and third bars, which are colored orange and black.

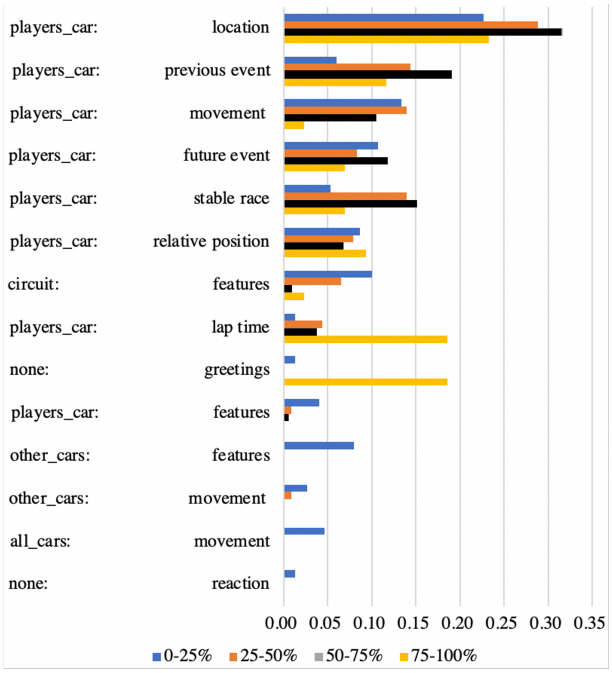


Figure 2: Distribution over utterance labels in different periods of timestamps

The proportion of *player's car:movement* in the second quarter (the second bars colored orange), is higher than in the third quarter (the third bars colored black). Thus, the commentators tend to mention more facts in the second quarter. In contrast, the third quarter (the third bars colored black) contains more *future event* and *previous event* labels that often include commentators' comments, concerns, or opinions on the previous and future events. This suggests that there are more mentions on the objective facts in the early stages of races, whereas subjective utterances increase toward the end of the races.

How do utterances differ depending on the viewpoint?

We examined the differences between commentaries from two different viewpoints: the driver's and aerial. Figure 3 shows the label distribution, where the upper bars colored orange correspond to the driver's viewpoint, and the lower bars colored blue correspond to the aerial viewpoint.

The proportion of the *player'car:location* for the aerial viewpoint is almost double of that of the proportion of the same label for driver's viewpoint. This is because the commentators with aerial viewpoint can capture the locations in maps more easily, whereas commentators with driver's

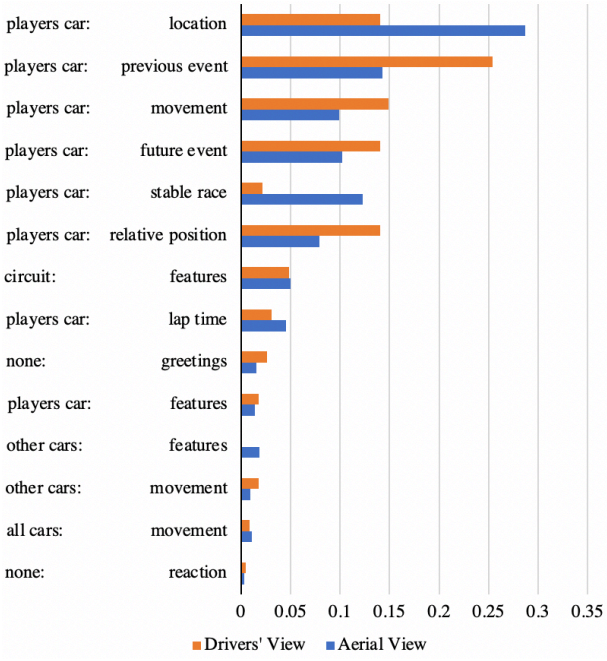


Figure 3: Distribution over utterance labels annotated from different viewpoints: driver's (the upper bars colored orange) or aerial (the lower bars colored blue).

viewpoint cannot see the entire circuit. Additionally, the proportion of *player's car:stable race* is very high for the aerial viewpoint. Because the aerial viewpoint is farther from the cars than the driver's viewpoint, the commentaries from the aerial viewpoint hardly mention slight movements of the cars; they more often say that the race is stable.

In contrast, the proportion of the *player's car:previous event* for the driver's viewpoint is higher than that of the aerial viewpoint. If the commentaries are spoken by the players themselves, they often comment on the events that just happened, e.g., *"OK, yes, the car turned successfully!"*.

The analysis above shows that the viewpoint influences the characteristics of the utterances.

4 Tasks and Models

We formulate a live commentary generation task and introduce the baseline models as shown in Figure 4. We report the performances of the baseline models for both subtasks to better understand the commentary generation task.

4.1 Task Formulation

To generate a live commentary, one needs to find multiple timepoints and generate an utterance at each timepoint. We solve this task in a sequential fashion; given the previous timepoint and its utterance, we find the next timepoint and generate its utterance, which will be solved below.

The task of timing identification is to determine the timestamp t at which an utterance should be generated. We formulate this problem as a binary classification for each second. Given the timepoint of the previously generated utterance, we iteratively classify each successive second according to whether the second is the next timepoint for generation or not. If the second is classified as positive, the model goes on to the generation step. If the second is classified as negative, the model goes on to the classification of the next second. If the model does not output positive for m seconds, the next second is forced to be positive. We set $m = 7$, which is double the average interval between two consecutive utterances.

For the classification of each second, we encode a given tuple (V, D, T). V denotes a sequence of the previous k video frames $V = (img_1, ..., img_k)$ captured every second. We set $k = 10$ in our experiments. We used `torchvision`[9] library to extract these images from videos. S denotes the structured data $D = \{D_1, ..., D_N\}$ consisting of N sets of the structured telemetry data, where each $D_n = \{val_{1,n}, ..., val_{k,n}\}$ consists of k values tracked at each of the previous k seconds. T represents the textual information, which is the previous utterance in our setting.

The task of the utterance generation is to generate a sequence of characters as an utterance, given the tuple of (V, D, T) for the given/estimated timepoint. In other words, we use the same information for both the second classification above and the utterance generation. We use a multimodal encoder-decoder architecture to generate an utterance.

4.2 Multi-modal Encoder

The models for both subtasks use the same network for encoding the input vision, structured telemetry and textual data. The encoded representation is then used in the network for subtasks. For video frames V, each video frame is converted to an image embedding by using Vision Transformer (Dosovitskiy et al., 2021)[10].

[9] https://pytorch.org/vision

[10] We used an open implementation at https://github.com/lucidrains/vit-pytorch.

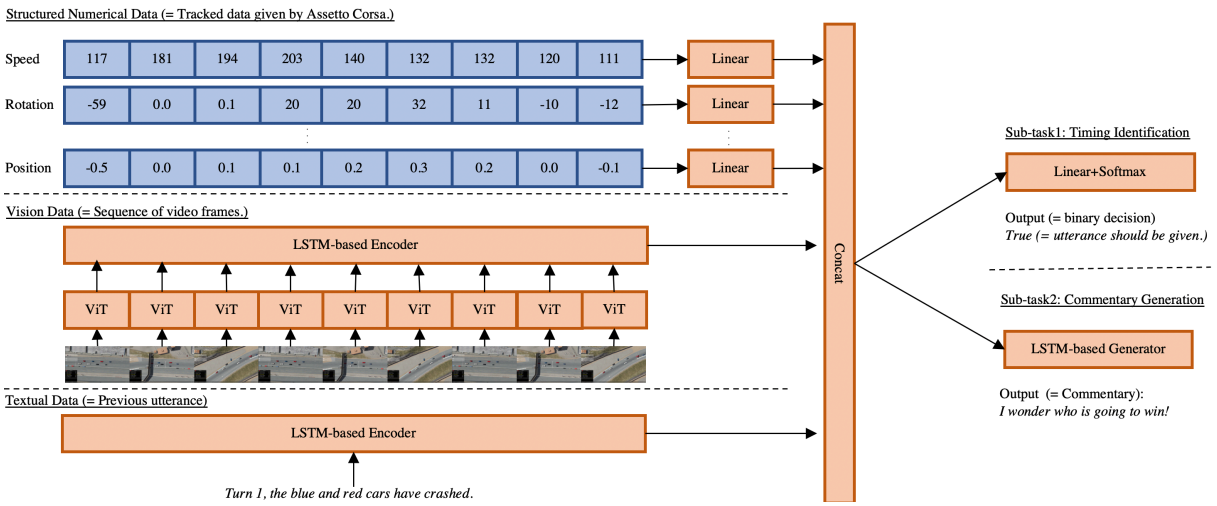


Figure 4: Baseline models for the timing identification and commentary generation tasks. Each sequence of numerical data i.e. speed, rotation, position and so on, is considered as a vector and we obtain a compressed vector. Vision information is encoded by using Vision Transformer and LSTM-based encoder. Textual information is encoded by using another LSTM-based encoder. The concatenated vector of the encoded numerical, vision and textual information is passed to the models for sub-tasks.

The image embeddings are then sequentially encoded by using a Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997): $h_{i,V} = LSTM_V(h_{i-1,V}, ViT(img_i))$, where `ViT` returns the output vector for the [CLS] token calculated by Vision Transformer. We treat the final state $h_{k,V}$ of the LSTM as the representation of V. For each sequence D_n of the structured data D tracked for the previous k seconds, we consider the sequence D_n as a vector for the n-th type of data in the structured telemetry data, and we transform it into an another representation by using a linear transformation: $d_n = ReLU(D_n W_d + b)$, where W_d is a weight matrix, b is a vector, and ReLU activates the vector. The concatenated vector of D_1 to D_N is the representation of D. For textual input, we simply embed characters in the textual input and then sequentially encode the embeddings by using another LSTM: $h_{i,T} = LSTM_T(h_{i-1,T}, emb_e(x_i))$, where emb_e returns the character embedding. We treat the final state of the LSTM as the representation for textual input T. Finally, the concatenated vector of the encoded representations of V, D, and T is passed to the networks for the sub-tasks explained next.

4.3 Timing Identification Model

For the timing identification, the encoded representation is passed to a network that consists of a linear transformation and the softmax function: $Softmax(encode([V; D; T])W_t)$, where $encode()$ returns the outputs of the encoder and W_t converts the concatenated vector to two-dimensional vector that represents the scores for the decisions to utter or not utter at this timepoint. We obtain the probability distribution over decisions by using the softmax function.

For training, we use the gold start timestamps from commentators as positive instances. We use the midpoint of the silence between consecutive utterances as negative instances. We train this classifier by using the cross-entropy loss. For testing, we classify every second after the time at which the previous utterance is given. We output the timestamp first classified as positive by our model.

4.4 Utterance Generation Model

This second subtask generates an utterance as a sequence of characters given an encoded representation V, S, and T. We use an encoder-decoder architecture with an attention mechanism, which consists of an LSTM-based decoder initialized by the representation passed from the encoder:

$$h_{j,d} = LSTM_d(h_{j-1,dec}, emb_d(y_{j-1})), \quad (1)$$

$$a_{ji} = \frac{exp(h_{j,d} W h_{i,V})}{\sum_{i=1}^{10} exp(h_{j,d} W h_{i,V})}, \quad (2)$$

$$c_j = \sum_i a_{ji} h_{i,V}, \quad (3)$$

$$o_j = Softmax([h_{j,d}; c_j] W_d), \quad (4)$$

where emb_d returns the embedding of a character,[11] c_j is a vector produced by an attention mechanism over the outputs of $LSTM_V$, and y_{j-1} is the previously generated character. W_d is a matrix that converts the concatenation $[h_{j,d}; c_j]$ to a vector of scores over the predefined vocabulary for the target utterances, and `Softmax` converts it to a probability distribution. This generator is trained by using cross-entropy loss.

5 Experiments

We conduct experiments for the two subtasks to further investigate the characteristics of the task.

5.1 Data and Parameters

We use 100 tuples of videos, commentaries, and structured data for validation, another 100 tuples for testing, and the remaining tuples for training. For Vision Transformer, we set the number of heads to six, the layer size to two, and each head is represented as a 100-dimensional vector. The parameter for the patch size is 30×30. The dropout rate was set to 0.1. Each type of telemetry data is represented as a 10-dimensional vector. We use three types of data i.e., speed, progress in a lap, steer rotation, and position on track. The dimensions of both the hidden states and input vectors to the LSTMs in encoders are set to 100. Thus, the dimension of the hidden state of the LSTM in the decoder side is 230, which is the sum of the size of the encoded images, textual information and structured data. The size of the character embeddings in the decoder is set to 100. We use separate vocabularies for the textual input and the target text. We use Adam (Kingma and Ba, 2015) with several initial learning rates ranging from 10^{-3} to 10^{-5} for optimizing parameters. We continue the training iterations until the loss in the validation dataset does not decrease for 10 epochs. We conduct the utterance generation experiments for the gold timestamps.

5.2 Timing Identification

We evaluate the models by using the average gaps in second between the gold timestamp and predicted timestamp. We propose a simple baseline that outputs the timestamp after 3.46 seconds from the end timestamp of the previous utterance. 3.46 is the average interval between two consecutive utterances as shown in Table 4. As a result, the average gap between the gold timestamps and predicted timestamps obtained from the baseline model was 3.66. When we use only structured data as input, we obtained the average gap of 3.27 seconds. Adding textual information achieved a slightly better value of 3.26, but the difference is negligible. Adding vision information improves the performance to 3.12.

[11] Note emb_d is different from emb_e.

Model	Avg. gap
baseline: average interval	3.66
struct	3.27
struct+text	3.26
struct+text+vision	**3.12**

Table 4: The average gap in seconds between the gold and predicted timestamps. Lower values are better.

Model	10^{-3}	10^{-4}	10^{-5}
struct	18.22	22.78	23.39
struct + text	18.03	23.78	23.86
struct + text + vision	17.49	22.58	**24.01**
only vision	0.30	2.74	7.46

Table 5: BLEU scores on the test dataset for the compared models trained on different learning rates. The model with the learning rate 10^{-5} achieves the best performance on the validation dataset.

5.3 Utterance Generation

We use BLEU (Papineni et al., 2002) to evaluate the baseline models for this task. The scores are shown in Table 5. The model based only on telemetry data worked well. Adding textual information improved BLEU score if the learning rate is set to lower values i.e., 10^{-4} or 10^{-5}. However, we obtained a very low BLEU score when we used only vision-based input. Adding vision information to struct+text model degraded the score if the learning rate is set to 10^{-3} or 10^{-4}. Even with a smaller learning rate, 10^{-5}, vision information did not significantly improve the performance.

6 Discussion

We list the gold and the utterances generated by the model with learning rate 10^{-4} in Table 6. Gold utterances often focus on relative position situations, as in Example 1, which requires capturing the physical relations between cars. However, as shown in the first example of a generated utterance by `data+text`, we found only a few generated utterances that mention the relative positions of

Example 1: timestamp: 00:55
Gold
The player is now following very close to the car ahead.
data+text
Now we're on Turn 10, the player is now accelerating
data+text+vision
I want to step on the brakes firmly here.
Example 2: timestamp: 02:04 and 02:07
Gold
We are now approaching the chicane on Turn 11 and 12. The player should properly use the curb and go on a straight line here, and the player showed stable race here.
data+text
The player should brake properly here. The player should brake properly here.

Table 6: The gold and automatically generated commentaries. Texts are translated from Japanese.

the player's car and other cars. Integrating vision information further reduces such utterances mentioning relative positions and other detailed information, and also makes utterances less specific. To generate utterances with detailed information, a model must accurately capture the information displayed in a small area of the image. However, it may be too hard for the model to, for example, capture the distance between the car driven by the player and the car just behind, or the drastic changes of speeds from the video frames shown in Figure 1, whereas telemetry data provides the useful information. From the perspective of studies on vision, methods to properly capture such features are worth exploring.

We also observed that generated commentaries contain many repetitions of the same utterance, especially utterances generated by the model with vision information. The utterances in Example 2 in Table 6 exemplifies repetitions. It should be note that the two utterances are only three seconds apart. The input to the model does not change significantly during such a short period of time, resulting in the two identical utterances. Some mechanisms to increase the diversity of utterances might alleviate this problem, which is a particular challenge in commentary generation.

We found errors in the name of a country e.g., *Nürburgring in Germany* was generated as *Nürburgring in Italy*. Such errors are also known as a common problem in other generation tasks.

7 Future Research Directions

Finally, we discuss the future directions. We noticed that evaluation is very difficult for this task. Only BLEU scores of course cannot capture the correctness because this evaluation ignores the relation between a commentary and a race represented in. However, manually checking videos, language, structured data, and generated utterances would incur a huge labor cost. An exploration into correct and efficient automatic and manual evaluation methods that consider all vision, language, and structured data should be conducted in the future. For evaluation by using BLEU, it might be helpful if we have multiple reference utterances for one timestamps. However, it is difficult to collect multiple utterances simultaneously in this task because different commentators give utterances at different timings. We leave them for an important future research direction.

Extensions of model would be considered as one of the main steps to produce better commentaries. However, more importantly, we need to explore an essential research question: "what is a good commentary?". Further analysis of the characteristics that contributes to making commentaries better need to be conducted.

8 Conclusion

In this paper, we proposed the task of generating commentaries for motor racing games. Our analysis reveals that the characteristics of utterances change over time in a race, and such changes are also caused by differences in viewpoints. They also show that combining vision, language and structured data is challenging, which worth studying in depth. For future work, exploring better methods to combine vision, language, and structured data will be a promising direction for future work. We release the data to enhance further studies on generation tasks from multimodal inputs.

Acknowledgements

This paper is based on results obtained from a project JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). For experiments, computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by National Institute of Advanced Industrial Science and Technology (AIST) was used. We thank KUNOS Simulazioni for granting us the permission to distribute our dataset of Assetto Corsa.

References

Kasumi Aoki, Akira Miyazawa, Tatsuya Ishigaki, Tatsuya Aoki, Hiroshi Noji, Keiichi Goshima, Hiroya Takamura, Yusuke Miyao, and Ichiro Kobayashi. 2021. Controlling contents in data-to-document generation with human-designed topic labels. *Computer Speech Language*, 66:101154.

Tatsuya Aoki, Akira Miyazawa, Tatsuya Ishigaki, Keiichi Goshima, Kasumi Aoki, Ichiro Kobayashi, Hiroya Takamura, and Yusuke Miyao. 2018. Generating market comments referring to external resources. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 135–139.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, pages 1–21.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Int. Res.*, 61(1):65–170.

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

Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoaki Okazaki, and Hiroya Takamura. 2019. Learning to select, track, and generate for data-to-text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2102–2113.

Byeong Jo Kim and Y. Choi. 2020. Automatic baseball commentary generation using deep learning. *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, page 1056–1065.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *International Conference on Computer Vision (ICCV)*, pages 706–715.

Mitsumasa Kubo, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2013. Generating live sports updates from twitter by finding good reporters. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 527–534.

Daniel Memmert and Dominik Raabe. 2018. *Data analytics in football: Positional data collection, modelling and analysis*. Routledge.

Soichiro Murakami, Sora Tanaka, Masatsugu Hangyo, Hidetaka Kamigaito, Kotaro Funakoshi, Hiroya Takamura, and Manabu Okumura. 2021. Generating weather comments from meteorological simulations. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Main Volume*, pages 1462–1473.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1374–1384.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Ratish Puduppully and Mirella Lapata. 2021. Data-to-text generation with macro planning. *Transactions of the Association for Computational Linguistics*, 9(0):510–527.

Michael Schaffrath. 2003. Mehr als 1:0! Bedeutung des Live-Kommentars bei Fußballubertragungen– eine explorative Fallstudie [more than 1:0! the importance of live commentary on football matches – an exploratory case study]. *Medien und Kommunikationswissenschaft*, 51.

Yasufumi Taniguchi, Yukun Feng, Hiroya Takamura, and Manabu Okumura. 2019. Generating live soccer-match commentary from play data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):7096–7103.

Yui Uehara, Tatsuya Ishigaki, Kasumi Aoki, Hiroshi Noji, Keiichi Goshima, Ichiro Kobayashi, Hiroya Takamura, and Yusuke Miyao. 2020. Learning with contrastive examples for data-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING2020)*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *2015 IEEE International*

Conference on Computer Vision (ICCV), pages 4507–4515.

Luowei Zhou, Chenliang Xu, and Jason J Corso. 2018. Towards automatic learning of procedures from web instructional videos. In *AAAI Conference on Artificial Intelligence*, pages 7590–7598.

Explaining Decision-Tree Predictions by Addressing Potential Conflicts between Predictions and Plausible Expectations

Sameen Maruf[†] **Ingrid Zukerman**[†] **Ehud Reiter**[‡] **Gholamreza Haffari**[†]
[†]Dept. of Data Science and AI, Faculty of IT, Monash University, Victoria, Australia
[‡]Dept. of Computing Science, University of Aberdeen, Scotland, UK
[†]{firstname.lastname}@monash.edu [‡]e.reiter@abdn.ac.uk

Abstract

We offer an approach to explain Decision Tree (DT) predictions by addressing potential conflicts between aspects of these predictions and plausible expectations licensed by background information. We define four types of conflicts, operationalize their identification, and specify explanatory schemas that address them. Our human evaluation focused on the effect of explanations on users' understanding of a DT's reasoning and their willingness to act on its predictions. The results show that (1) explanations that address potential conflicts are considered at least as good as baseline explanations that just follow a DT path; and (2) the conflict-based explanations are deemed especially valuable when users' expectations disagree with the DT's predictions.

1 Introduction

Machine Learning (ML) models have become increasingly accurate in recent times, leading to their widespread adoption by decision makers in a variety of vital domains, including healthcare, defense and energy. This underscores the need for explanations of the outcomes of these models that support decision making by practitioners.

ML models may be classified into transparent and opaque models based on their interpretability (Doshi-Velez and Kim, 2017). Transparent models are "interpretable by a Machine Learning expert or a statistician" (Biran and McKeown, 2017). These models, e.g., Decision Trees (DTs), decision rules and linear models, are built on the basis of interpretable features, which are typically obtained through feature engineering. Transparent models are often less accurate than opaque models, in particular neural networks, provided large training datasets are available. Nonetheless, it is necessary to explain transparent models because (1) large datasets may not always be available, as is the case in our evaluation datasets (§ 4.1); (2) it is common practice to clarify the outcomes of opaque models by approximating them with transparent models (§ 2); and (3) even if these transparent models are understandable by ML experts, they may still be unclear to practitioners.

In this paper, we generate textual explanations of predictions made by a particular transparent model: DT. Our explanations address potential conflicts between aspects of these predictions and plausible expectations licensed by background information (i.e., expectations that "make sense" in light of this information). Specifically, we identify four types of conflicts whereby events that appeared unlikely or likely on the basis of background information happened or did not happen respectively; we then specify schemas that address these conflicts (§ 3).

We generated explanations for two datasets: *Telecom* and *Nursery*. In Telecom, a DT predicts whether a customer will churn (leave) or stay with the company based on their profile (e.g., whether they have a phone service and what are their monthly charges); in Nursery, a DT predicts the acceptance status of a child to a childcare center on the basis of the circumstances of the child and their family (e.g., how satisfactory are the current childcare arrangements and how demanding is the parents' employment). The bottom part of Table 1 illustrates an explanation generated for an instance in the Nursery dataset. The explanation addresses a potential conflict between (a) a plausible expectation that a child with a good childcare situation is likely to be *Wait listed*, and (b) the DT's prediction that the child will be *Priority accepted*.

Our human evaluation of the explanations generated for the two datasets (§ 4) considers users' overall preferences for different explanation types, and the effect of explanations on two explanatory goals: users' understanding of the DT's reasoning, and

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 114–127,
Aberdeen, Scotland, UK, 20-24 September 2021.

Feature	Value
Parents' employment:	Challenging
Current childcare:	Good
Child's health:	Average

From the data, one might expect that children with good *current childcare* will be a great deal more likely to get *Wait listed* than to get a *Priority acceptance* (54% vs 11%). However, the AI system has learned from the data that among children with **challenging *parents' employment* and average *health***, those with good *current childcare* are almost certain to get a *Priority acceptance* (close to 100%).

Table 1: Explanation for the prediction of an instance in the Nursery dataset (bottom part); features used in the prediction and their values (top part).

their willingness to act on its predictions.[1] In addition, users rated the explanations on completeness, and on the presence of extraneous information.

The main findings of our user study are: (1) explanations that address potential conflicts are generally considered at least as good as baseline explanations that just follow a DT path; and (2) the conflict-based explanations are deemed especially valuable when users' expectations disagree with DT predictions. We stress that these findings pertain to explanations that address conflicts due to *plausible expectations* from background information. We do *not* claim that these explanations address *actual* user expectations.

2 Related Work

In 1990-2000, explanations derived from knowledge bases were enhanced by addressing aspects of users' reasoning. Specifically, Zukerman and McConachy (1993) and Horacek (1997) considered potential inferences from explanations, omitting easily inferable information and addressing erroneous inferences; Korb et al. (1997) took into account reasoning fallacies when explaining the reasoning of Bayesian Networks; and Stone (2000) generated instructions from which users could draw appropriate inferences about actions to take. Recently, Krause and Vossen (2020) identified additional triggers that should be addressed in explanations.

Current research on explanation generation focuses on explaining the predictions made by ML models – a sub-field called *Explainable AI* (*XAI*). In particular, neural networks have received a lot of attention owing to their superior performance on one hand, and their opaqueness on the other hand. A common first step in explaining the predictions of neural networks is to build a *local surrogate explainer model* that uses a transparent model to approximate the neighbourhood of an instance of interest. Linear regression (Ribeiro et al., 2016; Štrumbelj and Kononenko, 2014; Lundberg and Lee, 2017), decision rules (Ribeiro et al., 2018) and DTs (van der Waa et al., 2018; Guidotti et al., 2019; Sokol and Flach, 2020a) have been employed for this purpose.

A DT's prediction is generally explained by tracing the path from the root to a predicted outcome (Guidotti et al., 2019; Stepin et al., 2020). Recently, researchers have generated class-contrastive counterfactual explanations to enhance the explanations of DT predictions. Stepin et al. (2020) generated explanations that have a factual and a counterfactual component; the former is the DT trace, while the latter was found by ranking all the paths leading to alternative outcomes according to their distance from the factual explanation. Sokol and Flach (2020b) studied counterfactual explanations for DTs in an interactive system where users could change or remove features, or request an explanation for a hypothetical instance. Counterfactual explanations were generated by representing the tree structure as binary meta-features, and minimizing an $L1$-like metric to retrieve the shortest statement. However, these works do not determine when a counterfactual enhancement is required.

The need for an enhancement was studied in (Biran and McKeown, 2017) — they identified and addressed unexpected effects of individual features on predictions made by logistic regression. However, they did not consider unexpected predictions.

Reiter (2019) argued that good explanations must be written for a specific purpose and audience, have a narrative structure, and use vague language to communicate uncertainty. The explanations generated in (Sokol and Flach, 2020b) and (Biran and McKeown, 2017) have a narrative structure, and only those in (Biran and McKeown, 2017) use vague language to convey strength of evidence.

The approach described in this paper complements explanations by addressing both unexpected predictions and unexpected effects of features, thereby enhancing their narrative structure. In addition, we leverage the work of Elsaesser and Henrion (1989) to address Reiter's desideratum of using vague language to convey probabilities.

Finally, and more broadly, expectation-theory posits that the surprisingness of an event may stem

[1] The participants in our study were told that they have an AI, but they were not informed about the specifics of the ML model. Other explanatory objectives include enhancing trust in the system, and helping debug a system (Reiter, 2019).

from a discrepancy between the state of the world and propositions that are deducible from presented information (Ortony and Partridge, 1987). Itti and Baldi (2009) offer a Bayesian formulation of the influence of surprisingness on visual attention shifts in terms of the difference between prior and posterior probabilities. In our research, we employ a probabilistic formulation to identify potential conflicts between plausible expectations and aspects of DT predictions.

3 Justifying DT predictions

In this work, we explain the outcome predicted by a DT for sample instances, where an instance comprises a set of *features*, each associated with a *value*, and an outcome is a *discrete class*. For example, the top of Table 1 shows the features and values of a Nursery instance;[2] the DT then classifies this instance into one of three classes: *Reject*, *Wait list* and *Priority accept*.

Like Biran and McKeown's (2017) approach, ours hinges on identifying discrepancies, but it differs from their approach in that (1) we propose *addressing potential conflicts* as a guiding principle for selecting content that complements explanations of DT predictions; (2) these conflicts pertain to predicted outcomes and to the impact of variables; and (3) we identify these conflicts by comparing aspects of a DT prediction with plausible expectations derived from probabilistic relations.

3.1 Potential Conflicts

First, we define *potential conflicts*, and their building blocks: *plausible expectations* and *aspects of a DT prediction*. We then specify language-based probabilistic relations that are the basis for plausible expectations, and describe the identification of potential conflicts.

Plausible expectations pertain to the outcome and to the impact of a value j of feature x_i, denoted $x_{i,j}$. They are derived from prior and posterior probabilities of outcomes by means of relations R1-R3 and associated constraints (Table 2).

R1. *Posterior*$(\mathcal{C}|\, x_{i,j})$ vs *Prior*$(\mathcal{C})$

R2. *Posterior*$(\mathcal{C}'|\, x_{i,j})$ vs *Prior*$(\mathcal{C}')$

R3. *Posterior*$(\mathcal{C}'|\, x_{i,j})$ vs *Posterior*$(\mathcal{C}|\, x_{i,j})$

where *Prior*(c) is the prior probability of class c, *Posterior*$(c|x_{i,j})$ is the probability of class c given feature value $x_{i,j}$, $\mathcal{C}$ is the class predicted by a DT, and $\mathcal{C}'$ is an alternative class with the highest *Posterior* probability. The posterior probability of a class c is calculated from training data for each feature value $x_{i,j}$. If it is high, it licenses an expectation for $x_{i,j}$ to yield class c; and if it is low, the expectation is for $x_{i,j}$ *not* to yield class c. For example, if according to the data, children with ordinary parents' employment have a lower probability of getting a *Priority acceptance* to the childcare center than children in the general population (R1), it is plausible to expect a child with such parents' employment *not* to be *Priority accepted*.[3]

Aspects of a DT Prediction pertain to the class $\mathcal{C}$ *Predicted* by the DT, and the *Impact* of feature value $x_{i,j}$ on this class, denoted *Impact*$(x_{i,j}, \mathcal{C})$. *Impact* is TRUE if $x_{i,j}$ influences the *Predicted* class $\mathcal{C}$ — for a DT, this happens when $x_{i,j}$ is in the path to $\mathcal{C}$; *Impact* is FALSE otherwise.

A potential conflict takes place when an expected outcome differs from the class predicted by a DT (R4), or when a feature value that was expected to have an impact does not (R5).[4]

R4. *Plausible* outcome $\neq$ *Predicted* class $\mathcal{C}$

R5. *Plausible* impact of $x_{i,j} \neq$ *Impact*$(x_{i,j}, \mathcal{C})$

In our example, a potential conflict ensues because, contrary to the expectation, the class *Predicted* for the child is *Priority accept* (R4).

It is worth noting that relations R1-R3 and R4 are model agnostic: R1-R3 depend on probabilities obtained from the data, and R4 depends on R1-R3 and the *Predicted* class. However, the determination of the *Impact* of a variable in R5 depends on the model, e.g., as seen above, variable impact for DTs is determined by path membership.

The values of relations R1-R3 are obtained from discretized probabilistic relations (§ 3.1.1).

3.1.1 Discretizing probabilistic relations

To generate explanations that use language to communicate relative probabilities, we harness the research in (Elsaesser and Henrion, 1989), which

[2] Sample features for the evaluation datasets and their values appear in Table 4; the DT feature values for the Nursery dataset are described in Table 10, Appendix A.

[3] Our formalism assumes that users are aware of the prior and posterior probabilities of outcomes (they were given this information in our evaluation, § 4.2), and employs these probabilities as the basis for explaining DT predictions. Hence, it differs from probabilistic models, such as Bayesian Networks or Naïve Bayes, which use probabilities to infer outcomes.

[4] Biran and McKeown (2017) consider situations where a variable may be expected to have a high or a low impact. But in a probabilistic formulation, expecting an event with low probability is tantamount to expecting this event *not* to happen with high probability.

Conflict name	Relations licensing plausible expectations	R4		R5	
		***Plausible* outcome**	***Predicted* class**	***Plausible* impact of $x_{i,j}$**	***Impact*$(x_{i,j}, \mathcal{C})$**
Plausible$\neg\mathcal{C}$ / *Predict*$\mathcal{C}$	R1: $Post(\mathcal{C}\mid x_{i,j}) < , \simeq Prior(\mathcal{C})$ $Post(\mathcal{C}\mid x_{i,j}) < Post(\neg\mathcal{C}\mid x_{i,j})$	$\neg\mathcal{C}$	$\mathcal{C}$	TRUE	TRUE
—	R1: $Post(\mathcal{C}\mid x_{i,j}) > Prior(\mathcal{C})$	$\mathcal{C}$	$\mathcal{C}$	TRUE	TRUE
Plausible$\mathcal{C}$ / *Predict*$\mathcal{C}$-$x_{i,j}$*NoImpact*	$\forall C_k \neq \mathcal{C}\ Post(\mathcal{C}\mid x_{i,j}) > Post(C_k\mid x_{i,j})$ $\exists x_{m,n} \quad Post(\mathcal{C}\mid x_{i,j}) > Post(\mathcal{C}\mid x_{m,n})$				FALSE
Plausible$\mathcal{C}'$ / *Predict*$\mathcal{C}$ "vanilla"	R1: $Post(\mathcal{C}\mid x_{i,j}) < , \simeq Prior(\mathcal{C})$ R2: $Post(\mathcal{C}'\mid x_{i,j}) > Prior(\mathcal{C}')$	$\mathcal{C}'$	$\mathcal{C}$	TRUE	TRUE
Plausible$\mathcal{C}'$ / *Predict*$\mathcal{C}$-$x_{i,j}$*NoImpact*	R3: $Post(\mathcal{C}'\mid x_{i,j}) > Post(\mathcal{C}\mid x_{i,j})$ $\forall C_k \neq \mathcal{C}'\ Post(\mathcal{C}'\mid x_{i,j}) > Post(C_k\mid x_{i,j})$				FALSE

Table 2: Definition of potential conflicts (explanations appear in Tables 1 and 3): $\mathcal{C}$ denotes the *Predicted* class, and $\mathcal{C}'$ denotes an alternative class that has the highest *Posterior* probability (*Post* is shorthand for *Posterior*); the colours of (in)equalities match those in Figure 1; text in Column 4 indicates surprise about the plausible outcome in Column 3, and text in Column 6 expresses surprise about the plausible impact of $x_{i,j}$ in Column 5.

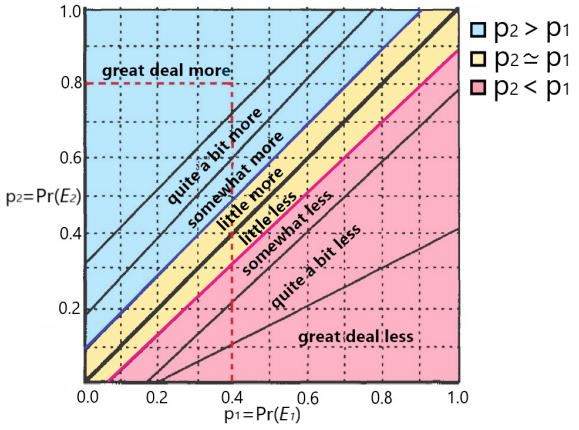


Figure 1: Verbal mapping of relative probabilities.

maps probability differences into verbal expressions. Figure 1 depicts their empirically derived phrase-selection function, which achieved a 72% accuracy compared to people's actual usage. For example, if the probability of event E_1 is $p_1 = 0.4$, and that of event E_2 is $p_2 = 0.8$ (dashed red lines in Figure 1), the phrase "E_2 is *a great deal more* likely than E_1" is selected.

Following a small pilot study to validate these expressions for our explanations, we merged the intermediate expressions "somewhat more/less" and "quite a bit more/less" in Figure 1 into simply "more/less". The resultant six-phrase mapping is used to define the wording for relations R1-R3.

3.1.2 Identifying Potential Conflicts

Table 2 displays the potential conflicts addressed by our explanations. Each segment represents a potential conflict, with the surprises boxed in red. Column 1 shows the name of the conflict, Column 2 displays the relations that license plausible expectations for an outcome and for the impact of feature value $x_{i,j}$ (the colour-coded relations are computed as specified in Figure 1, while the constraints are calculated using point probabilities); Column 3 presents the plausible expected *outcome* derived from the relations defining the conflict (Column 2); Column 4 shows the actual *Predicted* class $\mathcal{C}$; Column 5 displays the plausible expected *impact* of $x_{i,j}$ — a feature value that satisfies the relations defining a conflict (Column 2) is always expected to have an impact; and Column 6 shows the actual $Impact(x_{i,j}, \mathcal{C})$. Relation R4 is calculated by comparing the values of Columns 3 and 4, and Relation R5 is obtained from Columns 5 and 6.

We now describe each conflict illustrated with examples from the Nursery dataset.

***Plausible*$\neg\mathcal{C}$/*Predict*$\mathcal{C}$** (top segment in Table 2). This conflict arises when it is plausible to expect that in light of $x_{i,j}$, class $\mathcal{C}$ will not happen (Column 3), but surprisingly, $\mathcal{C}$ is *Predicted* (Column 4). The expectation is plausible because the posterior probability of class $\mathcal{C}$ given $x_{i,j}$ is less than or equal to its prior probability (R1), and also lower than the posterior probability of $\neg\mathcal{C}$ (Column 2). For this conflict, we only examined the case where $Impact(x_{i,j}, \mathcal{C})$=TRUE, i.e., $x_{i,j}$ is in the DT path. The FALSE case was disregarded, as the ensuing potential conflict seemed weak. However, for completeness, this case should be revisited in the future.

Example (full text in Table 3): In the Nursery dataset, children with critical *current childcare* are less likely to be *Wait listed* than applicants overall (R1: *Posterior* < *Prior*). However, in the context of other information about a particular child, having critical *current childcare* gets them *Wait listed* (R4: *Plausible* outcome $\neg\mathcal{C} \neq$ *Predicted* class $\mathcal{C}$).[5]

[5] As seen in Table 10, Appendix A, the term "critical childcare" indicates high insecurity in obtaining this service.

Schema	Sample Generated Explanations for the Nursery dataset
Conflict-based (outcome only): *Plausible¬C/PredictC*	
Preamble: $x^*_{i,j}$ + R1 + $\mathcal{C}$	From the data, one might expect that children with critical *current childcare* will be less likely than applicants overall to get *Wait listed* (19% vs 34%).
Resolution: ***Path*** + $x^*_{i,j}$ + $\mathcal{C}$	However, the AI system has learned from the data that among children with **ordinary *parents' employment*, somewhat problematic *social situation*** and **good *health***, those with critical *current childcare* are almost certain to get *Wait listed* (close to 100%).
Conflict-based (impact of feature value only): *PlausibleC/PredictC-$x_{i,j}$NoImpact*	
Preamble: $x^*_{i,j}$ + R1 + $\mathcal{C}$	From the data, one might expect that children with challenging *parents' employment* will be more likely than applicants overall to get a *Priority acceptance* (46% vs 32%).
Resolution: x^*_i + R5 + ***Path*** + $\mathcal{C}$	However, the AI system has learned from the data that the *parents' employment* has no effect on the outcome in this situation, and that children with **very critical *current childcare*** and **good *health*** are almost certain to get a *Priority acceptance* (close to 100%).
Conflict-based (outcome and impact of feature value): *PlausibleC′/PredictC-$x_{i,j}$NoImpact*	
Preamble: $x^*_{i,j}$ + R3 + $\mathcal{C}'$ + $\mathcal{C}$	From the data, one might expect that children with ordinary *parents' employment* will be more likely to get *Wait listed* than to get a *Priority acceptance* (47% vs 19%).
Resolution: x^*_i + R5 + ***Path*** + $\mathcal{C}$	However, the AI system has learned from the data that the *parents' employment* has no effect on the outcome in this situation, and that children with **very critical *current childcare*** and **average *health*** are almost certain to get a *Priority acceptance* (close to 100%).
Basic (no conflict): counterpart of *PlausibleC′/PredictC-$x_{i,j}$NoImpact*	
Path + $\mathcal{C}$	The AI system has learned from the data that children with **very critical *current childcare*** **and average *health*** are almost certain to get a *Priority acceptance* (close to 100%).

Table 3: Schemas that address three of the potential conflicts defined in Table 2 and Basic schema (our baseline), with sample explanations; relative probabilities are described in Figure 1; the selection of a *pivot feature* value is described in § 3.2; font denotes **feature values** and ***features*** in the DT path, and *Classes*.

PlausibleC/PredictC-$x_{i,j}$NoImpact (bottom of second segment in Table 2). This conflict occurs when a feature value $x_{i,j}$ is expected to have an impact (Column 5), but it has no effect on the *Predicted* class, i.e., it is not in the DT path (Column 6). The expectation for $x_{i,j}$ to have an impact arises when the posterior probability of class $\mathcal{C}$ in light of $x_{i,j}$ is higher than its prior probability (R1) and the posterior probabilities of all the other classes, and it is also higher than the posterior probability of class $\mathcal{C}$ in light of at least one other feature value in the current DT path — $x_{i,j}$ cannot be the "weakest" among the mentioned features (Column 2). Here, the plausible expectation for class $\mathcal{C}$ matches the DT's prediction, i.e., there is no conflict about the expected outcome.

Example (full text in Table 3): In the Nursery dataset, children with challenging *parents' employment* are more likely to get *Priority accepted* than the general population (R1: *Posterior* > *Prior*), but *parents' employment* is not in the DT path (R5: *Plausible* impact ≠ actual *Impact*).

PlausibleC′/PredictC (third segment in Table 2). Here, an alternative outcome $\mathcal{C}'$ is a plausible expectation from $x_{i,j}$ (Column 3), but surprisingly, class $\mathcal{C}$ is *Predicted* (Column 4). This conflict resembles *Plausible¬C/PredictC* in that the posterior probability of class $\mathcal{C}$ in light of $x_{i,j}$ is relatively low, i.e., $\neg\mathcal{C}$ is plausible (R1). However, *PlausibleC′/PredictC* goes further, nominating a potential alternative class $\mathcal{C}'$. The expectation for $\mathcal{C}'$ is plausible because its posterior probability is higher than its prior (R2) and the posterior of $\mathcal{C}$ (R3), and $\mathcal{C}'$ has the highest posterior probability among all the classes (Column 2). This conflict has two variants: "**vanilla**" – only the *Predicted* class is unexpected (top of the third segment); and ***$x_{i,j}$NoImpact*** – both the *Predicted* class and the lack of impact of $x_{i,j}$ (Column 6) are unexpected (bottom of the third segment).

Example of the first variant (full text in Table 1; the second variant appears in Table 3): In the Nursery dataset, children with good *current childcare* are more likely to get *Wait listed* than *Priority accepted* (R3: $Posterior(\mathcal{C}') > Posterior(\mathcal{C})$). However, a particular child with certain feature values and good *current childcare* gets *Priority accepted* (R4: *Plausible* outcome $\mathcal{C}' \neq$ *Predicted* class $\mathcal{C}$).

3.2 Generating Conflict-based Explanations

The inputs to the explanation generator are: an instance, a *Predicted* class and a set of conflicts. At present, our explanations address a potential conflict with respect to one feature value only.[6] Thus, for each conflict type, we first select a *pivot feature value* (denoted $x^*_{i,j}$), and then realize our explanation. We do not select a particular conflict type for an instance, as making this determination is one of the aims of our evaluation (§ 4.3.3).

[6] In the future, we will consider higher-dimensional spaces, which may require addressing several features with conflicts or adopting a different strategy, e.g., an interactive approach.

3.2.1 Selecting a pivot feature value

If several feature values qualify for a potential conflict type, we choose the strongest in terms of word mapping, e.g., "a great deal more" is stronger than "more". Ties are broken as follows: for *Plausible¬C/PredictC* and *PlausibleC/PredictC-$x_{i,j}$NoImpact*, we choose the $x^*_{i,j}$ with the maximum absolute difference between *Posterior*$(\mathcal{C}|\, x^*_{i,j})$ and *Prior*$(\mathcal{C})$ for the *Predicted* class $\mathcal{C}$. For the *PlausibleC′/PredictC* variants, we select the $x^*_{i,j}$ with the maximum difference between *Posterior*$(\mathcal{C}'|\, x^*_{i,j})$ and *Posterior*$(\mathcal{C}|\, x^*_{i,j})$.

3.2.2 Realizing explanations

A Conflict-based explanation has two main parts: *Preamble*, which presents a plausible expectation from the pivot feature value $x^*_{i,j}$; and *Resolution*, which describes how this expectation is thwarted. Table 3 displays schemas that address three potential conflicts, and one Basic schema (which is our baseline), together with sample explanations; an explanation that illustrates *PlausibleC′/PredictC* "vanilla" appears in Table 1 (the schema for this potential conflict is [*Preamble*: $x^*_{i,j}$ + R3 + $\mathcal{C}'$ + $\mathcal{C}$; *Resolution*: ***Path*** + $x^*_{i,j}$ + $\mathcal{C}$]). Since the focus of our research is on content selection, the explanations are realized by means of domain-independent programmable templates.

The ***Preamble*** presents probabilistic relations that license plausible expectations. The preambles of *Plausible¬C/PredictC* and *PlausibleC/PredictC-$x_{i,j}$NoImpact* describe relation R1; and those of the *PlausibleC′/PredictC* variants convey R3.

The ***Resolution*** has two components: (1) the feature values in the DT path that lead to the *Predicted* class $\mathcal{C}$, which also constitutes the Basic baseline explanation (Guidotti et al., 2019; Stepin et al., 2020); and (2) the impact of $x^*_{i,j}$, or lack thereof, in the context of the other feature values in the DT path. The features in the DT path are presented in a pre-established order (Table 4), except for $x^*_{i,j}$, whose placement is determined by the schemas: when $x^*_{i,j}$ is in the DT path, it appears right before the *Predicted* class; otherwise, the lack of impact of x^*_i is announced at the start of the *Resolution*.

4 Empirical Evaluation

Our evaluation considers two main questions: (Q1) How do Conflict-based explanations compare to Basic baseline explanations? (Q2) Which types of Conflict-based explanations are preferred to Basic explanations, if any?

Nursery	
Classes:	*Priority accept*, *Wait list*, *Reject*
parents' employment:	challenging, somewhat difficult, ordinary
current childcare:	very critical, critical, insufficient, sufficient, good
housing condition:	inadequate, somewhat inadequate, adequate
social situation:	problematic, somewhat problematic, unproblematic
child's health:	poor, average, good

Telecom			
Classes:	*Stay*, *Churn (leave the company)*		
senior citizen:	yes, no	***phone service***:	yes, no
internet service:	Fiber optic,	DSL,	no
online security:	yes,	NA (no internet service),	no
tenure (months with company):	1 month,		72 months
monthly charges:	\$19,		\$117

Table 4: Classes, sample features (in the presentation order used in our explanations) and values in the evaluation datasets; the feature values in the Nursery DT are described in Table 10, Appendix A.

Next, we describe our datasets and classifier, followed by our experimental design and results.[7]

4.1 Datasets

We used two datasets, which were pre-processed as described in Appendix A: *Nursery* (Olave et al., 1989), which has 12630 instances and three classes; and *Telecom*, which has 3302 instances and two classes. These datasets were chosen due to their diverse character, and the differences in number and types of features and predicted classes. Both datasets were split into 80% training and 20% test sets using proportional sampling (we did not cross-validate, as average classifier accuracy is tangential to this research).

We employed the J48 classifier (Quinlan, 1993) in WEKA (Frank et al., 2016) to learn DTs. It produced a DT with 47 nodes for the Nursery dataset (93% accuracy on the test set) and a DT with 41 nodes for Telecom (80% accuracy on the test set).[8] 78% of the Nursery test samples and all the Telecom test samples had at least one potential conflict.

4.2 Experiment Design

Our experiment starts with a demographic questionnaire followed by the body of the survey.

The body of the survey begins with a narrative immersion, where participants are told that they are the director of a childcare center (Nursery) or the sales representative of a telecommunications company (Telecom), and that they have purchased an AI system to help them predict the acceptance status of prospective pupils (Nursery) or whether cus-

[7] We have addressed the recommendations for human evaluation in (Howcroft et al., 2020). The experiment and data are available at https://doi.org/10.26180/15147462.

[8] Users are informed of a DT's overall accuracy, but not about its accuracy for individual predictions — in the future we will study the inclusion of this information in an explanation.

tomers will churn (leave) or stay (Telecom). The participants are then given a brief account of how an AI makes predictions, and shown the features and values that are input to the AI (illustrated in Table 4) — a screenshot of the introductory narrative for the Nursery dataset appears in Figure 2, Appendix D. Next, a sequence of scenarios is presented in random order, each pertaining to a different family/customer — a screenshot of a Nursery scenario appears in Figure 3, Appendix D. Between scenarios, a short version of the Matching Familiar Figures Test (MFFT) (Cairns and Cammock, 1978) is shown as a filler.

Scenario description. We chose scenarios with the strongest available potential conflict (using a procedure similar to that in § 3.2.1), and diverse pivot and explanatory variables. Scenarios without conflicts were excluded from our evaluation, as they warrant only a Basic explanation. To ensure that all the potential conflicts in Table 2 are represented, we chose eight Nursery scenarios (four each for *Wait list* and *Priority accept*)[9] and ten Telecom scenarios (five each for *Stay* and *Churn*).

Each scenario begins by showing a set of features such as those in Table 4, together with their values for a particular family/customer and the *Prior* and *Posterior* probabilities of the possible classes. Users are then asked to make an educated guess about the predicted class, after which they are shown the prediction made by the DT.

Next, users are given two side-by-side explanations for this prediction: Conflict-based versus Basic. The selection of a side (left or right) for an explanation type is randomized between scenarios, but all the participants see the same side-by-side configuration for a given scenario.

Users' views about explanations. Users are then asked to enter their level of agreement on a 5-point Likert scale ('Strongly disagree':1 to 'Strongly agree':5) with statements about four explanatory attributes: completeness of an explanation and presence of misleading/contradictory/irrelevant information, as well as the understandability of the AI's reasoning and their willingness to act on the prediction on the basis of an explanation (exact statements appear in the screenshot in Figure 3, Appendix D). The first three attributes come from Hoffman *et al.*'s (2018) *Explanation Satisfaction Scale*, and the third and fourth attributes are our explanatory goals (§ 1). Participants are also asked which explanation(s) they prefer, if any.

To detect unreliable responses, we inserted an attention question, where we asked users to indicate whether a neutral statement about the background information in the scenario was true or false.

[9]Examples for *Reject* were not presented, as there was only one reason to reject applicants, viz poor health.

Question	Option	Nursery	Telecom
Gender	Male / Female	12 / 28	25 / 17
Age	18-34 years old	33	37
Ethnicity	Asian / Caucasian	17 / 17	28 / 4
English proficiency	Medium / High	5 / 36	5 / 37
Education	Bachelor / Master	13 / 13	14 / 22
ML expertise	Low / Med-High	27 / 14	18 / 24
Domain familiarity	Yes / No	9 / 32	31 / 11

Table 5: Descriptive statistics: for gender, age, ethnicity and education, we present the options that had most participants; domain familiarity was self-rated.

Participant cohorts. To avoid participant fatigue, we conducted a separate experiment for each dataset — details appear in Appendix B. The surveys were implemented in the Qualtrics survey software, and conducted on SONA.

We obtained a total of 83 valid responses out of 109 — 41 for Nursery and 42 for Telecom (responses were validated based on the answers to the attention questions and the total time spent on the experiment). Table 5 shows the statistics for the Nursery and the Telecom cohorts.

4.3 Results

To answer Q1, we compared Conflict-based explanations with Basic ones for each dataset in terms of the four explanatory attributes mentioned above, and user preferences (§ 4.3.1). We also analyzed the influence of various independent variables on users' ratings of Conflict-based explanations compared to Basic ones (§ 4.3.2). To answer Q2, we analyzed how individual Conflict-based explanations compare to their Basic counterparts (§ 4.3.3).

Statistical significance for the ratings of the four attributes for Conflict-based versus Basic explanations was obtained using Wilcoxon signed-rank test; a one- and two-proportion Z-test was respectively used for the proportion of preference counts within one population and between two populations. Statistical significances were adjusted with Holm-Bonferroni correction for multiple comparisons (Holm, 1979).

4.3.1 Conflict-based vs Basic explanations

Our results show that for the Nursery dataset (top of Table 6), Conflict-based explanations are deemed

Attribute	Conflict-based Mean (SD)	Basic Mean (SD)	Stat. Sig.
Nursery			
Complete	3.43 (0.97)	3.00 (0.98)	< 0.001
Misleading...	2.72 (1.00)	2.55 (0.89)	< 0.05
Understandable	3.61 (1.04)	3.02 (1.03)	< 0.001
Willingness to act	3.56 (1.01)	3.23 (1.01)	< 0.001
Telecom			
Complete	3.22 (0.99)	2.93 (0.97)	< 0.001
Misleading...	3.00 (1.14)	2.81 (1.05)	–
Understandable	3.49 (0.92)	3.33 (0.87)	–
Willingness to act	3.16 (0.99)	3.09 (0.94)	–

Table 6: Comparison between explanation types: scores and statistical significances (Wilcoxon signed-rank test); a lower score is better for Misleading... , and a higher score is better for the other attributes.

	Count					χ^2	Stat. Sig.
	Conflict-based	Basic	Both	None	Total		
Nursery	112	45	13	35	205	28.59	< 0.001
Telecom	117	78	11	46	252	7.80	< 0.01

Table 7: Preference for an explanation type: χ^2 statistic and statistical significances (one-proportion Z-test) calculated from clear preferences for Conflict-based/ Basic explanations.

significantly more complete, understandable and enticing to act on a DT's prediction than Basic explanations. However, Conflict-based explanations are also deemed more misleading/contradictory/irrelevant than Basic explanations. For Telecom (bottom of Table 6), Conflict-based explanations are considered significantly more complete than Basic explanations, but equivalent for the other three attributes.

In terms of preferences, for both datasets, the majority of users prefer Conflict-based explanations to Basic ones (Table 7). However, the two datasets differ significantly in the proportions of preferences for Conflict-based explanations (two-proportion Z-test, *p-value* < 0.05; proportions calculated from the data in Table 7), with a higher percentage of users preferring the Conflict-based explanations for the Nursery dataset.

4.3.2 Influence of independent variables

Our experiment has several independent variables, including predicted outcome, pivot feature, explanation length and (dis)agreement between an expected and a predicted class. The first two variables are scenario-specific, and hence offer no opportunities to draw generalizable conclusions.

Regarding explanation length, Lombrozo (2016) reported that users generally prefer longer explanations, in particular when they include jargon. However, in our case, length is highly correlated with explanation type — Conflict-based explanations have 60 words on average in both Nursery and Telecom, and Basic explanations have 29 words. Hence, we cannot analyze length separately from explanation type. Nonetheless, our results suggest that length cannot be the only factor influencing users' views, as some types of Conflict-based explanations have similar preferences to Basic explanations (Table 9).

Attribute	Predict vs Expect	Conflict-based Mean (SD)	Basic Mean (SD)	Stat. Sig.
Nursery				
Complete	Pred=Exp	3.41 (0.96)	3.04 (0.97)	< 0.01
	Pred≠Exp	3.48 (0.99)	2.90 (0.99)	< 0.01
Misleading...	Pred=Exp	2.80 (1.03)	2.54 (0.90)	< 0.05
	Pred≠Exp	2.57 (0.92)	2.57 (0.86)	–
Understand-able	Pred=Exp	3.61 (1.07)	3.20 (0.99)	< 0.01
	Pred≠Exp	3.61 (0.97)	2.66 (1.01)	< 0.001
Willingness to act	Pred=Exp	3.64 (0.95)	3.41 (0.98)	< 0.05
	Pred≠Exp	3.40 (1.12)	2.87 (0.98)	< 0.01
Telecom				
Complete	Pred=Exp	3.18 (0.97)	2.99 (0.95)	–
	Pred≠Exp	3.35 (1.04)	2.72 (1.01)	< 0.01
Misleading...	Pred=Exp	3.08 (1.14)	2.83 (1.05)	–
	Pred≠Exp	2.75 (1.10)	2.75 (1.08)	–
Understand-able	Pred=Exp	3.45 (0.90)	3.35 (0.86)	–
	Pred≠Exp	3.62 (0.98)	3.25 (0.93)	–
Willingness to act	Pred=Exp	3.14 (0.97)	3.17 (0.90)	–
	Pred≠Exp	3.25 (1.07)	2.83 (1.04)	< 0.05

Table 8: Effect of (dis)agreement between users' expectations and DT predictions: scores and statistical significances (Wilcoxon signed-rank test).

Interestingly, our analysis shows that (dis)agreement between users' expectations *according to their survey answers* and the class *Predicted* by the DT has a significant influence on the ratings of Conflict-based explanations compared to Basic ones (users' answers disagreed with a *Predicted* class when they selected a different class or *Can't Decide* – see options in Figure 3, Appendix D).

For the Nursery dataset, the general results obtained for Conflict-based versus Basic explanations hold for completeness, understandability and willingness to act on predictions for both agreement and disagreement between users' expectations and DT predictions (top of Table 8). However, Conflict-based explanations were deemed to contain more misleading/contradictory/irrelevant information than Basic ones only when users' expectations matched DT predictions. This suggests that the additional information provided by Conflict-based explanations is welcome when a prediction is *not* as expected.

For the Telecom dataset, Conflict-based explanations were considered more complete and enticing

Basic vs Conflict-based	Nursery							Telecom						
	Count					χ^2	Stat. Sig.	Count					χ^2	Stat. Sig.
	Conflict	Basic	Both	None	Total			Conflict	Basic	Both	None	Total		
Basic vs *Plausible¬C/PredictC*	33	12	3	14	62	9.80	< 0.01	46	21	2	15	84	9.33	< 0.01
Basic vs *PlausibleC/PredictC-$x_{i,j}$NoImp*	8	6	1	6	21	0.29	–	14	20	2	6	42	1.06	–
Basic vs *PlausibleC′/PredictC* "vanilla"	33	13	6	9	61	8.70	< 0.01	23	6	3	10	42	8.53	< 0.05
Basic vs *PlausibleC′/PredictC-$x_{i,j}$NoImp*	38	14	3	6	61	11.08	< 0.01	34	31	4	15	84	0.14	–

Table 9: Preference for individual explanation types: χ^2 statistic and statistical significances (one-proportion Z-test) calculated from clear preferences for Conflict-based/Basic explanations (*NoImp* is shorthand for *No Impact*).

to act only when users' expectations differed from DT predictions (bottom of Table 8).

In terms of preferences, most users preferred Conflict-based explanations to Basic ones for the Nursery dataset, regardless of the agreement between users' expectations and DT predictions (*p-value* < 0.001, Table 12 in Appendix C). However, for Telecom, Conflict-based explanations were preferred only when users' expectations disagreed with DT predictions (*p-value* < 0.001).

4.3.3 Individual Conflict-based explanations

Our comparison between individual Conflict-based explanations and their Basic counterparts shows that a statistically significantly higher proportion of users preferred *Plausible¬C/PredictC* and *PlausibleC′/PredictC* "vanilla" to Basic explanations for both Nursery and Telecom (Table 9). But *PlausibleC′/PredictC-$x_{i,j}$NoImpact* was preferred to its Basic counterpart only for the Nursery dataset, where it had the largest margin. Finally, *PlausibleC/PredictC-$x_{i,j}$NoImpact*, which addresses a conflict with respect to variable impact only, was deemed equivalent to its Basic counterpart for both datasets. However, according to (Biran and McKeown, 2017), users were more satisfied with explanations about unexpected variable impacts than no explanation. This suggests that further studies are required to determine the conditions for explaining unexpected variable impacts.

The results in Table 9 indicate that if a DT prediction has several qualifying conflicts, they should be prioritized in the following order: *Plausible¬C/PredictC* $\succ$ *PlausibleC′/PredictC* "vanilla" $\succ$ *PlausibleC′/PredictC-$x_{i,j}$NoImpact*.

5 Conclusion

Our approach for explaining DT predictions addresses potential conflicts between aspects of these predictions and plausible expectations licensed by background information. To this effect it operationalizes the identification of four types of conflicts, and specifies schemas for generating explanations that address these conflicts. Our approach is model agnostic, except for the determination of the actual impact of a variable, which is readily available in most ML models.

Our evaluation on the Nursery and Telecom datasets shows that (1) explanations addressing potential conflicts between DT predictions and plausible expectations from background information are considered at least as good as baseline explanations; and (2) the Conflict-based explanations are deemed especially valuable when users' expectations disagree with DT predictions.

These insights are of practical import, since users' expectations are often not available to explanation systems, and Conflict-based explanations provide clear benefits, or at worst are neutral, regardless of the particulars of these expectations.

Our approach has the following limitations, which we propose to address in the future: (1) it does not perform feature selection to reduce long paths in a DT; (2) Conflict-based explanations address only one pivot feature; and (3) the explanations omit information about DT accuracy for particular instances.

Our evaluation has the following limitations: (1) we cannot divorce length from explanation type, as Conflict-based explanations are about twice as long as Basic ones; (2) the cohorts for the two datasets had different demographics, so, given the size of our population, it is not possible to attribute differences in our results for each dataset to domain or demographic differences; and (3) we could not recruit participants with relevant experience, but in light of our narrative immersion and the general accessibility of the concepts in the explanations, we believe that our results are informative.

Acknowledgments

This research was supported in part by grant DP190100006 from the Australian Research Council. We thank Marko Bohanec, one of the creators of the Nursery dataset, for helping us understand the features and their values. We also thank the anonymous reviewers for their helpful comments.

References

Or Biran and Kathleen McKeown. 2017. Human-centric justification of Machine Learning predictions. In *IJCAI'17*, pages 1461–1467, Melbourne, Australia.

Ed Cairns and Tommy Cammock. 1978. Development of a more reliable version of the Matching Familiar Figures test. *Developmental Psychology*, 14(5):555–560.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable Machine Learning. Arxiv:1702.08608.

Christopher Elsaesser and Max Henrion. 1989. Verbal expressions for probability updates: How much more probable is "much more probable"? In *UAI'89*, pages 319–330, Windsor, Canada.

Eibe Frank, Mark A. Hall, and Ian H. Witten. 2016. The WEKA workbench. In *Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques (Fourth ed.)"*. Morgan Kaufmann Publishers, San Francisco, California.

Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. 2019. Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems*, 34(6):14–23.

Robert R. Hoffman, Shane T. Mueller, Gary Klein, and Jordan Litman. 2018. Metrics for explainable AI: Challenges and prospects. Arxiv:1812.04608.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

Helmut Horacek. 1997. A model for adapting explanations to the user's likely inferences. *User Modeling and User-Adapted Interaction*, 7(1):1–55.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *INLG2020*, pages 169–182, Dublin, Ireland.

Laurent Itti and Pierre Baldi. 2009. Bayesian surprise attracts human attention. *Vision Research*, 49(10):1295–1306.

Kevin B. Korb, Richard McConachy, and Ingrid Zukerman. 1997. A cognitive model of argumentation. In *CogSci 1997*, pages 400–405, Stanford, California.

Lea Krause and Piek Vossen. 2020. When to explain: Identifying explanation triggers in human-agent interaction. In *NL4XAI'2020*, pages 55–60, Dublin, Ireland.

Tania Lombrozo. 2016. Explanatory preferences shape learning and inference. *Trends in Cognitive Sciences*, 20(10):748–759.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *NIPS'17*, pages 4768–4777, Long Beach, California.

Manuel Olave, Vladislav Rajkovic, and Marko Bohanec. 1989. An application for admission in public school systems. In I.Th.M. Snellen, W.B.H.J. van de Donk, and J.-P. Baquiast, editors, *Expert Systems in Public Administration*, chapter 10, pages 145–160. Elsevier.

Andrew Ortony and Derek Partridge. 1987. Surprisingness and expectation failure: What's the difference? In *IJCAI'87*, page 106–108, Milan, Italy.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Francisco, California.

Ehud Reiter. 2019. Natural language generation challenges for explainable AI. In *NL4XAI'2019*, pages 3–7, Tokyo, Japan.

Marco T. Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?": Explaining the predictions of any classifier. In *KDD'16*, pages 1135–1144, San Francisco, California.

Marco T. Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *AAAI-18*, pages 1527–1535, New Orleans, Louisiana.

Kacper Sokol and Peter Flach. 2020a. LIME-tree: Interactively customisable explanations based on local surrogate multi-output regression trees. Arxiv:2005.01427.

Kacper Sokol and Peter Flach. 2020b. One explanation does not fit all: The promise of interactive explanations for Machine Learning transparency. Arxiv:2001.09734.

Ilia Stepin, Jose M. Alonso, Alejandro Catala, and Martin Pereira. 2020. Generation and evaluation of factual and counterfactual explanations for decision trees and fuzzy rule-based classifiers. In *WCCI*, pages 1–8, Glasgow, Scotland.

Matthew Stone. 2000. Towards a computational account of knowledge, action and inference in instructions. *Journal of Language and Computation*, 1:231–246.

Erik Štrumbelj and Igor Kononenko. 2014. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665.

Jasper van der Waa, Marcel Robeer, Jurriaan van Diggelen, Matthieu Brinkhuis, and Mark Neerincx. 2018. Contrastive explanations with local Foil Trees. In *WHI'2018*, pages 41–46, Stockholm, Sweden.

Ingrid Zukerman and Richard McConachy. 1993. Generating concise discourse that addresses a user's inferences. In *IJCAI'93*, pages 1202–1207, Chambery, France.

A Datasets

Feature value	Description
	Parents' employment
challenging	frequent relocations, transfers, long leaves of absence; parents are not employed in the school district and need to travel more than one hour for work.
somewhat difficult	hard working conditions that allow for an early retirement (e.g., miners, policemen, soldiers), night work, additional work engagements.
ordinary	normal condition.
	Current childcare
very critical	there is no possibility of childcare with family, and previous level of childcare was inadequate (child does not live with parents, problematic private care).
critical	there is no possibility of childcare with family, and previous level of care was less than adequate (frequent change of care, termination of care, alternate care by parents, occasional care).
insufficient	no possibility of childcare with family (both parents or single parent work full-time or are full-time students, no alternative care with relatives), but previous level of care was adequate (with own family, adequate private care, educational care organizations).
sufficient	childcare is possible with some relatives (healthy and unemployed grandparents living in the school district, other able-bodied and unemployed members of the household).
good	normal condition (childcare is possible in the family – father or mother unemployed and able to care).
	Housing condition
inadequate	subleased or emergency housing; cramped; has lack of sanitation facilities or water.
somewhat inadequate	subleased or cramped apartment.
adequate	normal condition.
	Social situation
problematic	inadequate educational ability of parents (gross neglect of education and care, violence); inadequate family relationships (serious conflicts between parents, between grandparents, between parents and grandparents, more severe forms of disturbance of parents or other family members); social and antisocial forms of restraining behavior by parents and other family members (alcoholism and other addictions, delinquency, quitting, etc).
somewhat problematic	less than adequate educational ability of parents (uneven, inconsistent education, excessive difficulty or indulgence, neurotic reaction of parents); less than adequate family relationships (milder forms of parental personality disorders, privileged or neglected children, family conflicts).
unproblematic	normal condition.
	Child's health
poor	admission is not recommended due to the health conditions of the child.
average	the child has a mental or physical disorder that influences their admission status; the child's development is affected by health conditions of family members.
good	normal condition (healthy).

Table 10: Description of feature values in the Nursery DT; all the feature values for *current childcare*, *housing condition*, *social situation* and *child's health*, except the value defined as normal, require the opinion of relevant professional services.

The Nursery dataset originally had five classes, three of which account for about 97% of the instances; we therefore removed the other two classes, which resulted in a balanced dataset with 12630 instances. The classes, features and feature values in the dataset were originally in Slovenian, and their English translation in (Olave et al., 1989) was somewhat peculiar. With the help of one of the authors of the original paper, we recoded the features and feature values in the Nursery domain to those in Table 4, and the names of the retained classes to *Reject*, *Wait list* and *Priority accept*. The recoded feature values are described in Table 10.

The Telecom dataset had only two classes, *Stay* and *Churn*, but it was imbalanced towards *Stay* (73%). The DT had an accuracy of 79% when trained with a cost-sensitive setting for imbalanced datasets. This accuracy is comparable to those reported in Kaggle for several predictive models. However, in order to avoid biasing participants' class expectations, we decided to even out the class distribution. To this effect, we retained only customers with a month-to-month contract, which had both outcomes, and randomly removed half of the incorrectly predicted cases. This yielded a more balanced dataset (60% *Stay*) and a slightly improved DT accuracy of 80% (trained without the cost-sensitive setting).

Table 11 shows final classes in the two datasets and the breakdown of the training/test sets.

Partition	**Nursery**				**Telecom**		
	Reject	*Wait list*	*Priority accept*	**Total**	*Stay*	*Churn*	**Total**
Training	3485	3414	3205	10104	1596	1057	2653
Testing	835	852	839	2526	390	259	649
Total	4320	4266	4044	12630	1986	1316	3302

Table 11: Breakdown of classes for the training set and the test set for the Nursery and Telecom datasets.

B Experiment Design

The scenarios studied in this paper compare Conflict-based explanations with Basic explanations for two datasets. However, our experiment contains additional scenarios, which compare two Conflict-based explanations. To limit the duration of an experiment to less than 1 hour, the experiment for each dataset was split into two parts — each part was shown to a different group of participants.

- Each Nursery group was shown five scenarios that compare Conflict-based explanations with Basic explanations, and two scenarios that compare two Conflict-based explanations; two of the former scenarios were common to both Nursery groups.
- Each Telecom group was shown six scenarios that compare Conflict-based explanations with Basic explanations, and one scenario that compares two Conflict-based explanations; as for Nursery, two of the former scenarios were common to both groups.

The common scenarios were used to determine whether the two participant groups for a particular dataset behave similarly. To this effect, we performed a two-proportion Z-test on preference for Conflict-based explanations in the common scenarios; we found no statistically significant differences between the preferences of the two Nursery groups (*p-value* $= 0.714$) or the preferences of the two Telecom groups (*p-value* $= 0.388$).

C Results

	Predict vs Expect	**Count**					χ^2	**Stat. Sig.**
		Conflict-based	Basic	Both	None	Total		
Nursery	Pred = Exp	74	35	9	20	138	13.95	< 0.001
	Pred ≠ Exp	38	10	4	15	67	16.33	< 0.001
Telecom	Pred = Exp	78	72	8	34	192	0.24	–
	Pred ≠ Exp	39	6	3	12	60	24.20	< 0.001

Table 12: Preferences broken up by (dis)agreement between users' expectations and DT predictions: χ^2 statistic and statistical significances (one-proportion Z-test) calculated from clear preferences for Conflict-based/Basic explanations.

D Screenshots from the Nursery survey

Background

We are developing a computer system that automatically generates explanations for predictions made by an Artificial Intelligence (AI) system. For example, say we have an AI system that predicts whether an applicant to a childcare centre will be accepted or rejected. Our explanation system generates several alternative explanations for this prediction.

The objective of this study is to find out which types of explanations people find useful in order to understand and accept the predictions of the AI system. We would appreciate your help in making this determination.

About the survey

In this survey, you will see seven situations together with some background information. We will present the outcome predicted by the AI system for each situation, and show you two alternative explanations for each outcome. You will then rate each explanation based on several criteria, such as clarity and completeness.

This experiment focuses on the childcare domain. We will first introduce you to this domain, and then we will give you an example of the questions you will get in the survey.

The childcare domain

You are the director of the Bilby Childcare Centre, a non-profit organisation whose aim is to serve all members of the community. Part of your job is to evaluate applications from the parents of prospective pupils. Evaluating these applications involves weighing the childcare needs of families across several factors, such as housing condition and health (see the table below), in order to accept children in most need of childcare. In the past, an admissions committee performed these assessments.

To make the admission process more efficient, you have purchased a state-of-the-art AI system that predicts the outcome of an application from the data considered by the committee and the decisions made by the committee in the past -- the possible outcomes are: **priority accept**, **wait-list** or **reject**. The accuracy of your AI system in predicting the committee's decisions is 93%.

AI systems make predictions based on trends and patterns they identify in the data. Therefore, they may determine that attributes that are relevant to some situations are not relevant to other situations. For example, if the family's current childcare arrangements are deemed 'sufficient', their housing condition may influence the AI system's prediction about the outcome of their application. In contrast, the AI system may not need to consider the housing condition, if the current childcare arrangements are deemed 'very critical'.

Each **applicant to the Bilby Childcare Centre** fills out an application form, which is transcribed into **five** factors that make sense to the AI system. The factors and their possible values are listed below in shades of red and blue. These colours will be used in the situations you will see in the survey.

Factor	Possible values				
Parents' employment	Challenging	Somewhat difficult	Ordinary		
Current childcare	Very critical	Critical	Insufficient	Sufficient	Good
Housing condition	Inadequate	Somewhat inadequate	Adequate		
Social situation	Problematic	Somewhat problematic	Unproblematic		
Health (of the child)	Poor	Average	Good		

Note: since the Bilby Childcare Centre is a community service, it is not equipped to serve children with poor *health*. Therefore, children with poor *health* are rejected, even if their other factors would normally warrant acceptance.

In the following pages, you will see seven applications to the Bilby Childcare Centre. For each application, we will:

- present the above factors and their values, together with a few general facts regarding these factors -- the factors and their values are used by the AI system to make its predictions;
- ask you to make an educated guess about the outcome of the application;
- show you the prediction made by the AI system, together with two alternative explanations for this prediction; and
- ask you to rate these explanations along several criteria, such as clarity and completeness. Your ratings should be informed by your role **as the director of the childcare centre**.

Before we proceed, let's look at a sample application and the questions you will be asked.

Figure 2: Narrative immersion for the Nursery survey.

Applicant Nicholson:
The Nicholson family has submitted an application for admission of their child to the Bilby Childcare Centre. Based on their responses in the application form, the factors and values in the first two columns in the table below have been entered into the AI system. The outcome statistics that pertain to the situation of the Nicholson family appear in the third, fourth and fifth columns.

Factor	Value	Outcome		
		Reject	Wait-list	Priority accept
Parents' employment	Challenging	34%	20%	46%
Current childcare	Sufficient	35%	55%	10%
Housing condition	Adequate	35%	40%	25%
Social situation	Unproblematic	35%	37%	28%
Health (of the child)	Average	0%	43%	57%

In general, 32% of the applicants are given Priority acceptance, 34% are Wait-listed, and 34% are Rejected.

As the director of the Bilby Childcare Centre, what is your expectation regarding the outcome of the Nicholsons' application given their situation and the above mentioned facts?

- ◯ Priority accept
- ◯ Wait-list
- ◯ Reject
- ◯ Can't decide (no particular expectation)

Our explanation system has produced two alternative explanations for this outcome.

With reference to Explanation A and Explanation B, indicate the extent to which you agree with the statements below in your role **as director of the childcare centre**.

Explanation A

From the data, one might expect that children with challenging *parents' employment* will be more likely to get a *Priority acceptance* than to get *Wait-listed* (46% vs 20%).

However, the AI system has learned from the data that among children with

- sufficient *current childcare*,
- adequate *housing condition* and
- average *health*,

those with challenging *parents' employment* are almost certain to get *Wait-listed* (close to 100%).

Recall that based on what it has learned from the data, the AI system may deem some factors to be irrelevant when predicting the outcome for a particular situation.

Explanation B

The AI system has learned from the data that children with

- challenging *parents' employment*,
- sufficient *current childcare*,
- adequate *housing condition* and
- average *health*

are almost certain to get *Wait-listed* (close to 100%).

Recall that based on what it has learned from the data, the AI system may deem some factors to be irrelevant when predicting the outcome for a particular situation.

	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree
This explanation helps me understand the reasoning of the AI system.	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
This explanation has misleading, contradictory or irrelevant information.	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
This explanation is complete (it is not missing information).	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Based on the explanation, I would perform the action predicted by the AI system.	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

According to the background information of the Nicholsons, indicate whether the following statement is True or False:

28% of the applicants with unproblematic social situation get a Priority acceptance.

- ◯ True
- ◯ False

As the director of the Bilby Childcare Centre, please indicate your opinion about the explanations.

- ◯ I prefer Explanation A
- ◯ I prefer Explanation B
- ◯ I like both explanations equally
- ◯ I don't like any of the explanations

Which factors did you consider important when determining your expectation about the outcome of the Nicholsons' application? Select all that apply.

Parents' employment	Current childcare	Housing condition	Social situation	Health	None apply
☐	☐	☐	☐	☐	☐

We would appreciate your suggestions about improving the explanations.

Figure 3: Background information about the Nicholson family scenario; question about the expected outcome; model prediction (displayed after an outcome has been selected); *PlausibleC′/PredictC* explanation "vanilla" (A) and Basic explanation (B) for this scenario; attention question; preferences for explanations; features that determine expectations; request for suggestions.

Formulating Neural Sentence Ordering as the Asymmetric Traveling Salesman Problem

Vishal Keswani
Indian Institute of Technology, Kanpur
vkeswani@iitk.ac.in

Harsh Jhamtani
Carnegie Mellon University
jharsh@cs.cmu.edu

Abstract

The task of Sentence Ordering refers to rearranging a set of given sentences in a coherent ordering. Prior work (Prabhumoye et al., 2020) models this as an optimal graph traversal (with sentences as nodes, and edges as local constraints) using topological sorting. However, such an approach has major limitations – it cannot handle the presence of cycles in the resulting graphs and considers only the binary presence/absence of edges rather than a more granular score. In this work, we propose an alternate formulation of this task as a classic combinatorial optimization problem popular as the Traveling Salesman Problem (or TSP in short). Compared to the previous approach of using topological sorting, our proposed technique gracefully handles the presence of cycles and is more expressive since it takes into account real-valued constraint/edge scores rather than just the presence/absence of edges. Our experiments demonstrate improved handling of such cyclic cases in resulting graphs. Additionally, we highlight how model accuracy can be sensitive to the ordering of input sentences when using such graphs-based formulations. Finally, we note that our approach requires only lightweight fine-tuning of a classification layer built on pre-trained BERT sentence encoder to identify local relationships.

1 Introduction

A logical and coherent structure is an important characteristic of easily comprehensible text. As such, modeling the structure of coherent texts has been an important problem in NLP (Barzilay and Elhadad, 2002). In this paper, we work on the task of sentence ordering, wherein given an unordered set of sentences, the aim is to generate the most coherent ordering among them (Table 1). It essentially arises in situations where the information available in parts (sentences) is to be presented as a whole in a coherent and logical ordering. Correctly ordering sentences has applications for summarization (Barzilay and Elhadad, 2002; Nallapati et al., 2017; Narayan et al., 2018), constructing natural language explanations (Jhamtani and Clark, 2020), automatic scoring of an essay (Attali and Burstein, 2006; Farag et al., 2018; Amorim et al., 2018), and automatic generation and evaluation of a narrative (See et al., 2019; Jhamtani and Berg-Kirkpatrick, 2020; Gangal et al., 2021; Sakaguchi et al., 2021).

Input	s_1: Our son really likes his new bike.
	s_2: It's almost Christmas time.
	s_3: They really love what they got.
	s_4: We had a very fun time.
	s_5: The Children begin to open their presents.
Correct Output Sequence: $s_2 \rightarrow s_5 \rightarrow s_3 \rightarrow s_1 \rightarrow s_4$	

Table 1: An instance of the Sentence Order Generation Task. Given a set of incoherently arranged sentences as input, the goal is to output a coherent ordering. (Story from the SIND dataset)

Much prior work has formulated sentence ordering as a sequence prediction task (Gong et al., 2016; Cui et al., 2020), by first encoding the input sentences, and then predicting the ordering or numbering of the sentences. There have also been attempts to model the task as a ranking problem (Kumar et al., 2020). More recently, Prabhumoye et al. (2020) model this as a constraint-solving problem, wherein they first identify the relative ordering between pairs of sentences using a BERT-based (Devlin et al., 2019) classifier. Thereafter, they formulate it as performing a topological sort on a graph, wherein each sentence is a node in a graph, and each directed edge represents a pairwise constraint. Compared to prior work, it simplifies the decoding process by using a graph traversal formulation. However, the topological sort algorithm used for translating constraints into the final order

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 128–139,
Aberdeen, Scotland, UK, 20-24 September 2021.

leads to major limitations. Firstly, it cannot handle cycles in the resulting graph, and picks some arbitrary ordering in such cases (Figure 1). Our analysis shows that cycles were present in more than 54% of the graphs across the four datasets under consideration. Secondly, though the underlying classifier can provide more fine-grained scores, the scores are discretized/binarized to run the topological sorting algorithm. This leads to a loss of useful information in the decoding process.

In this work, we propose a reformulation of Neural Sentence Ordering as the classical Traveling Salesman Problem, while leveraging the recent progress in large pre-trained models such as BERT. We build upon the classification model used in Prabhumoye et al. (2020) and overcome the limitations in their approach by making better use of the information yielded by the classifier, taking into account global dependencies by employing a combinatorial minimization objective, and working with an overall framework that can handle cycles in the resulting graph.

Our contributions can be summarized as follows. Firstly, we provide a novel formulation of the sentence ordering task as the Traveling Salesman problem. Compared to the previous graph-based approach of using topological sorting, our proposed technique gracefully handles the presence of cyclic constraints. Moreover, it is more expressive since it admits real-valued soft constraints as opposed to hard binary constraints. Secondly, experiments with multiple datasets demonstrate improved results under some setups compared to the baselines using alternative graph-based formulation. Finally, we observe how certain choices in data pre-processing in graph-based approaches for sentence ordering can affect accuracy scores. We propose and use a more robust data processing and evaluation. The code is publicly available. [1]

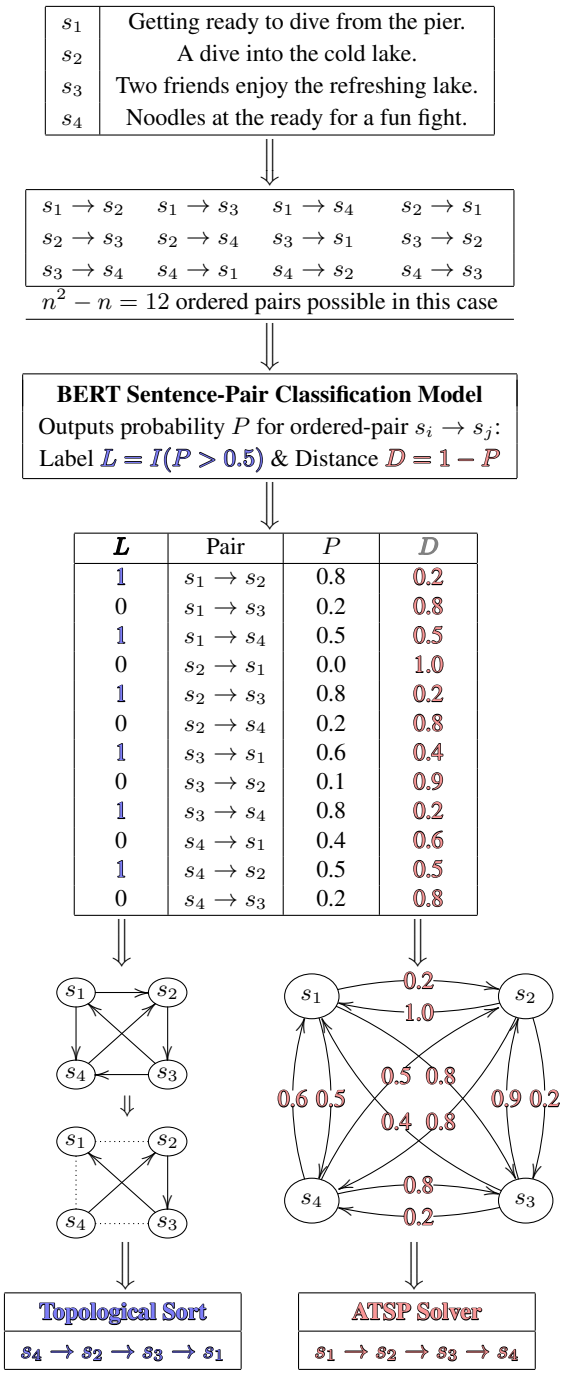


Figure 1: **Graph-traversal based formulation for sentence ordering task:** Such approaches first compute local pairwise constraints using next sentence prediction probability from a fine-tuned BERT classifier. **(a)** Topological Sort (Prabhumoye et al., 2020) discretizes the edges (0/1) and then runs topological sorting to get the final output sequence. However, such an approach is likely to pick an arbitrary ordering in the case of cycles. **(b)** In the proposed Traveling Salesman **ATSP** formulation, classifier probabilities are used to derive soft constraint scores between pairs of nodes, thus making use of more expressive fine-grained scores.

2 Background

In this section, we first formally describe the task of Sentence Ordering. Then we discuss its formulation as a constrained graph traversal, and discuss limitations of prior formulations of the task as a graph traversal problem.

2.1 Problem Formulation

Consider an 'unordered' set of n sentences: $S = \{s_1, s_2, ..., s_n\}$. Our aim is to find a permutation

[1] https://github.com/vkeswani/BerTSP

$P = \{i_1 i_2 ... i_n\}$ such that the resulting 'ordered' sequence is $S^* = \{s_{i_1}, s_{i_2}, ..., s_{i_n}\}$ is coherent.

Formulation as a Graph Traversal: Prabhumoye et al. (2020) propose to first identify binary constraints between all pairs of sentences. More specifically, given a pair of sentences $\{s_i, s_j\}$, they aim to extract whether s_i should follow s_j or the other way around. Thereafter, they decode the global ordering by treating the decoding as topological sorting in a graph, wherein sentences are treated as nodes, and pairwise constraints denote presence/absence of edges (i.e. there is an edge from s_i to s_j if a constraint states that s_j should follow s_i.). A high-level outline of this approach is shown in Figure 1.

2.2 Topological Sorting and its Limitations

To the best of our knowledge, (Prabhumoye et al., 2020) is the only work till now that utilizes a graph based formulation of Sentence Order Prediction on top of pairwise scores from a BERT-based classifier. Though it succeeds in achieving a light and efficient method for this task with minimal training, there are some inherent issues in this approach which we discuss below. Next, we will describe these limitations.

Discretization of edges: Prior work utilizes a BERT based classifier to predict local ordering constraints between pairs of sentences. However, to run the topological sorting, the classifier probability is converted a $0/1$ prediction, which governs merely the direction of the edge i.e. the fine-grained probability scores are thereafter not used. This leads to loss of valuable information about the likelihood of the edge, thus making it a significant issue. Since only binary constraints are learned and no score is attached to any order, it misses out on learning a rich global structure.

Cyclic constraints: The Topological Sort algorithm inherently cannot deal with cycles as it operates only on DAGs (Directed Acyclic Graphs). To deal with such cases, one can pick an arbitrary ordering of the nodes and edges, and delete edges that result in any cycles. However, such an approach will pick random orderings at best. For example, in Figure 1, the resulting graph has cycles, and the final output is a random ordering among the nodes.

Neutral pairs: To determine the direction of the edge between a pair of nodes, prior work feeds the pair of sentences to the classifier either as $s_1 \rightarrow s_2$ or as $s_2 \rightarrow s_1$ (both are equally likely). For instance, if $s_1 \rightarrow s_2$ is selected, the classifier predicts $P(s_1 \rightarrow s_2)$. If $P(s_1 \rightarrow s_2) > 0.5$, then the edge is directed from s_1 to s_2 in the graph (i.e. $s_1 \rightarrow s_2$). Otherwise, it is directed from s_2 to s_1 (i.e. $s_2 \rightarrow s_1$). Due to the lack of information in some sentence pairs or the limited efficacy of the classifier, the left out possibility is also probable. In such cases, both $P(s_1 \rightarrow s_2) > 0.5$ and $P(s_2 \rightarrow s_1) > 0.5$ are possible. We call such pairs neutral. The baseline approach does not consider breaking ties for neutral pairs. Datasets contain up to 50% samples with one or more neutral pairs. Note that $s_1 \rightarrow s_2$ and $s_2 \rightarrow s_1$ are not complementary events. Both lead to different input representations being fed to the classifier giving rise to different outputs (Section 3.3).

3 Sentence Ordering as the Traveling Salesman Problem

As mentioned previously, our objective is to work with soft constraints rather than binary constraints. To enable the use of such soft constraints, we cast the sentence ordering task as a Traveling Salesman traversal with respect to the graph denoting sentences as nodes, and constraints as edges. In the rest of this section, we first briefly describe the Traveling Salesman Problem (TSP) (Section 3.1), then describe how we reduce the sentences ordering as TSP (Section 3.2). Thereafter, we discuss the procedure to identify soft constraints using a classifier built on BERT representations. Finally, we discuss the solutions to solve TSP given a graph (Section 3.4).

3.1 Traveling Salesman Problem

The Traveling Salesman Problem is one of the more well-known problems studied in combinatorial optimization. In terms of graph theory, given an undirected weighted graph, it aims to find the shortest Hamiltonian Cycle, i.e. the cycle with the least weight that visits each node of the graph exactly once. Additionally, for our purpose and other practical applications, the graph is complete. Formally, we are given a complete undirected weighted graph $G = (V, E, W)$ where V denotes the set of vertices, E denotes the set of edges and W denotes the matrix containing weights for every edge.

$V = \{v_i\}_{\forall i \in \{1,2,...,n\}}$
$E = \{e_{ij}\}_{\forall i,j \in \{1,2,...,n\}}$

$W = \{w_{ij}\}\ \forall i,j \in \{1,2,...,n\}$

For G and a source vertex v_s ($s \in \{1, 2, ..., n\}$), we need to find a cyclic permutation or a Hamiltonian cycle $P = \{si_1i_2...i_{n-1}\}$ for which the following summation is minimized:

$$w_{si_1} + w_{i_1i_2} + ... + w_{i_{n-2}i_{n-1}} + w_{i_{n-1}s}$$

Since the permutation is cyclic, the summation is independent of the choice of s. This is the formulation of symmetric TSP for which $w_{ij} = w_{ji}$, i.e. distance from v_i to v_j is exactly the same as the distance from v_j to v_i. However, for some practical applications including ours, the two distances may not be equal. This is the case of asymmetric TSP.

3.2 Asymmetric TSP and Sentence Ordering

In the Asymmetric formulation of the Traveling Salesman Problem, $w_{ij} \neq w_{ji}$. These cases may arise in case of one-way traffic, accidental blockage, etc. The graph is still complete but directed with two edges between each pair of vertices in either direction. For the purpose of Sentence Ordering, we train a classifier that learns the probability of sentence i being followed by sentence j in the correct order and vice-versa. These probabilities give us the weights for the two edges between nodes v_i and v_j. Since the traditional TSP is proposed as a distance/cost minimization problem, a high probability should correspond to low distance. Hence, we use the probability of the complement (i.e. $1 - P$) as the weight.

$$w_{ij} = 1 - P(s_i \rightarrow s_j)$$
$$w_{ji} = 1 - P(s_j \rightarrow s_i)$$

Since the ground truth for the classifier is 1 for a correct pair-order and 0 for an incorrect pair-order, it is reflected in the predicted probabilities and hence, $P(s_i \rightarrow s_j) \neq P(s_j \rightarrow s_i)$. Intuitively, if sentence i is followed by sentence j in the correct order, the distance from i to j should be less than the distance from j to i, i.e. $w_{ij} < w_{ji}$. TSP requires the source to be pre-specified, for which we introduce a dummy node (Additional description about dummy node can be found in Appendix).

This way, each sentence serves as a vertex of graph G and the probabilities serve as entries of the weight matrix W. We use W to find the exact or heuristic solutions of the asymmetric TSP which eventually results in the correct Sentence Ordering.

In the TSP formulation, all of the issues with topological sorting are overcome. Firstly, the solution for TSP is independent of the order in which the input sentences are fed, hence preventing any sensitivity to the order in which sentences are processed. The graph constructed for every sample is cyclic and it does not have an inherent issue with cycles. This is reflected in its performance on samples with cycle(s) (table 2). Secondly, in the case of asymmetric TSP, weighted edges in both directions are required. This way both $P(s_1 \rightarrow s_2)$ and $P(s_2 \rightarrow s_1)$ make it to the weight matrix and the ambiguity of neutral pairs (local relationship) is resolved via their dependencies with other nodes (global relationship). Thirdly, since exact probability values are used as weights to arrive at a final order, there is no loss of information via discretization. Lastly, the order with the minimum cost arrangement is chosen which takes care of the global context. We refer to our method as **BerTSP**.

3.3 Learning Soft Constraints

We leverage BERT (Devlin et al., 2019) base-uncased configuration to obtain sentence pair representations. Specifically, we train a multi-layer feed-forward neural network that operates on sentence representation from pre-trained BERT, and makes a binary prediction about the relative ordering of the sentence pair. For every pair, it gives the probability of the first sentence (s_i) being followed by the second sentence (s_j). This way we obtain $P(s_i \rightarrow s_j)$ and consequently w_{ij}. It differs from (Prabhumoye et al., 2020) as it makes a prediction for either direction for a sentence pair ($s_i \rightarrow s_j$ and $s_j \rightarrow s_i$) while Prabhumoye et al. (2020) pick any one direction with a probability of 0.5 and use the binary label to define a constraint (unweighted edge). This way, for a set of n sentences, we obtain $2 \times C_2^n = n^2 - n$ scores which serve as off-diagonal elements of the distance matrix (diagonal elements are set to 0 as $P(s_i \rightarrow s_i) = 0$). This matrix is augmented with a row and a column of 0s to account for the dummy node discussed in the previous section and appendix. This augmented matrix serves as input for TSP.

In practice, we use the 'BertForSequenceClassification' module (Wolf et al., 2019) which takes the following sequence as input $[t_1^i, t_2^i, ..., t_n^i, \text{'SEP'}, t_1^j, t_2^j, ..., t_m^j]$. Here, the t^is represent tokens of sentence i and t^js represent tokens of sentence j. 'SEP' represents the separator which is a special token used by BERT.

3.4 Solution of ATSP

To produce an ordered sequence from the learned weight matrices, we delve into exact and approxi-

mate solutions to the Asymmetric TSP. The exact solution involves starting from the source node and calculating the cost of all permutations for the remaining nodes. This method is straightforward but expensive with a runtime complexity of $O(n!)$, where n is the number of sentences in the paragraph. Note that the above-mentioned complexity is only for the decoding process – the soft constraint computation involves running BERT only once per sentence i.e. number of forward passes on BERT scales linearly with the number of sentences ($O(n)$).

Approximation Algorithm for solving TSP: The popular approximate solvers available for TSP pose some limitations for our problem. They mainly focus on minimizing cost without regard to order. Their efficacy relies on the fraction of cost added to the optimal cost. They require assumptions such as triangle inequality. Hence, we consider a lightweight heuristic solution for the TSP problem where we simply sort the nodes (sentences) in the increasing order of the sum of soft-constraint scores arising at a given node. For example, for i^{th} node, the computed score is $\sum_{j}^{j!=i} P(s_i \rightarrow s_j)$. (Additional details about the employed approximation are provided in the appendix.) Since this process involves simple sorting, its runtime complexity for n sentences is $O(nlogn)$ (Same as the worst-case complexity for the topological sorting). We refer to this version of our method as **BerTSP-Approx**.

Ensemble Approach: Since the exact solution is expensive but generally more accurate while the above-mentioned approximation is computationally much cheaper but slightly less accurate in practice, we use a combination of these two as per the following criteria: We use the exact solution for samples with up to 10 sentences and the heuristic approximation for samples with more than 10 sentences. This gives us a practical solution that is feasible and almost optimal. We refer to this version of our method as **BerTSP-Ensemble**.

4 Experiments

4.1 Datasets

Stories: We use the textual portion of the Sequential Image Narrative Dataset or SIND (Huang et al.) which has been used in most of the previous studies on Sentence Ordering. It contains stories (image captions) with 5 sentences each.

Abstracts: We experiment on three research paper abstracts datasets, namely NIPS, AAN, and NSF, commonly used for this task. These are derived from NIPS conference papers, ACL Anthology Network corpus and NSF research award abstract dataset respectively (Logeswaran et al., 2018). (See appendix for data-split information.)

4.2 Evaluation

We employ the following five metrics to evaluate our approach. For all these metrics, a higher value corresponds to better performance.

Perfect Match Ratio (PMR) (Chen et al., 2016) measures the percentage of predicted orders that exactly match the correct order. It does not discount for minor differences.

Sentence Accuracy (ACC) (Logeswaran et al., 2018) measures the number of sentences having correct absolute positions in the predicted order. It is less strict than PMR.

Kendall's Tau (TAU) (Lapata, 2003) or Tau accounts for the pairwise relative order of sentences. Tau=1-2I/T where I represents the number of incorrect pair-orders in the predicted order and T represents total number of pair-orders in the correct order.

Rouge-S (R-S) (Chen et al., 2016) measures the number of skip-bigrams with the correct relative ordering in the predicted order as a percentage of the total number of skip-bigrams in the correct order. Here, skip-bigrams are pairs of sentences which may or may not be consecutive in the respective orders.

Longest Common Subsequence (LCS) (Gong et al., 2016) calculates the percentage of the longest common subsequence (not necessarily consecutive) between the predicted and the correct orders.

4.3 Call for Careful Data Pre-processing

We observe a potential risk of ground truth label leakage when employing graph-based methods. More specifically, if indexing of nodes (sentences) is performed in the same ordering as ground truth sequence, then the graph algorithms can inadvertently exploit this information. We demonstrate this through an illustrated example in Figure 2, wherein we show how the results of topological sorting change when indexing order of nodes is changed. In other words, if the input sentences

Figure 2: **Need for careful data pre-processing and evaluation:** When dealing with cycles in a graph, consider an implementation to handle cycles that always traverses nodes in the order $t = \{0, 1, 2, 3, 4\}$. For every node $j \in t$, cycle detection begins from j and if an edge $s_k \rightarrow s_j$ is encountered, it is deleted. As the traversal is in ascending order, for every deleted edge $s_k \rightarrow s_j$ $j < k$. Hence, edges that oppose the ascending order (t) are deleted and t is always favored. **(A)** In this case, the correct order is the same as t. Since t is favored, the predicted order is correct. **(B)** In this case, the correct order is the reverse of t (the rest of graph e is equivalent to a). Since t is favored, the predicted order is incorrect.

are indexed/fed as per the ground truth ordering, there is an information leak from the index numbers. To analyze the potential impact of such a pre-processing issue, we feed the input in the exact reverse of the correct order $\{n-1, n-2, ..., 0\}$, and present corresponding results for B-TSort (Prabhumoye et al., 2020) in Table 2. We observe that the accuracy values for methods like B-TSort can vary a lot based on the index ordering. From a practical use perspective, we are interested in analyzing the worst-case results across orderings. So from this point onward, we report results for B-TSort considering worst performance across orderings.

4.4 Results

We experiment with both variants of the proposed method. Additionally, we consider B-TSort method (Prabhumoye et al., 2020) as a baseline. (To the best of our knowledge, Prabhumoye et al. (2020) is the only work till now that delves into Graph Theory for Sentence Order Prediction like us.). We do not compare against non-graph based formulations since our aim is to improve upon the limitations of prior graph traversal based formulations.

We present the performance of B-TSort and BerTSP on the subsets of data that have cycles (as per discrete edge-based graphs used in B-TSort) in Table 2. The results demonstrate that the output of B-TSort varies as per the index ordering. Since the variation in the 3 presented outputs is very high, we prefer to use the worst-case output with BerTSP instead of the average case, henceforth. As shown in the table, we obtain significant worst-case improvements for both BerTSP-Approx and BerTSP-Ensemble, particularly the Perfect Match Ratio.

In Table 3, we present the results for whole

		PMR	ACC	TAU	R-S	LCS	PMR	ACC	TAU	R-S	LCS
		SIND - 12.74%					**AAN - 12.84%**				
B-TSort	Correct	26.71	55.37	0.64	81.88	78.97	28.87	57.42	0.79	87.54	81.18
	Shuffled	12.08	39.39	0.42	70.94	70.91	11.79	43.66	0.65	81.98	74.87
	Reverse	0.00	27.42	0.18	59.05	63.63	0.00	33.01	0.52	76.35	70.22
BerTSP	Approx	7.14	38.70	0.41	70.78	68.82	6.84	42.74	0.63	80.56	71.79
	Ensemble	12.42	39.19	0.41	70.62	71.06	12.50	43.26	0.64	80.89	74.50
		NIPS - 17.41%					**NSF - 70.47%**				
B-TSort	Correct	25.71	54.64	0.78	87.45	80.18	4.44	31.72	0.65	82.02	66.71
	Shuffled	12.00	43.50	0.67	82.97	74.71	1.61	22.99	0.51	76.18	60.64
	Reverse	0.00	33.93	0.56	78.01	70.54	0.00	19.18	0.37	70.30	58.05
BerTSP	Approx	10.00	44.11	0.64	82.24	71.96	0.75	20.98	0.45	71.76	54.23
	Ensemble	14.29	43.39	0.64	81.48	72.32	1.65	20.92	0.44	71.49	54.76

Table 2: **Results for data subset containing cycles as per graph representations used in B-TSort**: Percentage of cyclic cases and comparison of performance when using topological sorting on cyclic cases for three orderings in which inputs are fed: first is the correct order $\{0, 1, ..., n-1\}$, second is randomly shuffled order, and the third is the reverse of the correct order, i.e. $\{n-1, n-2, ..., 0\}$. The results demonstrate that if not properly handled, method outputs can depend highly on the indexing order of nodes. The two variants of BerTSP improve significantly on the reverse ordering output as depicted in the table.

datasets, comparing B-TSort and BerTSP on the worst case. We observe that BerTSP outperforms B-TSort on all metrics across all datasets (except LCS on NSF dataset). The improvements are also statistically significant with $p < 0.05$. It achieves up to 20% improvement on Perfect Match Ratio, 11% on Kendall Tau and 6% on position-wise Sentence Accuracy. It also improves on Rouge-S and LCS. Since these two metrics do not penalize gaps, their values are on the higher end and consequently, the improvements are less as compared to the other metrics. We also provide some qualitative examples in the appendix.

Model	**PMR**	**ACC**	**TAU**	**R-S**	**LCS**
		SIND			
B-TSort	16.28	48.29	0.54	77.04	75.08
BerTSP-Approx	17.01	49.96	0.57	78.65	75.60
BerTSP-Ensemble	19.54	49.75	0.56	78.11	76.35
		NIPS			
B-TSort	27.36	56.42	0.77	86.91	81.01
BerTSP-Approx	32.09	59.86	0.79	88.27	81.75
BerTSP-Ensemble	32.59	59.36	0.79	87.75	81.79
		AAN			
B-TSort	46.67	64.54	0.79	86.61	83.81
BerTSP-Approx	48.09	66.44	0.81	87.81	84.19
BerTSP-Ensemble	48.81	66.41	0.81	87.59	84.57
		NSF			
B-TSort	6.84	24.22	0.45	71.32	61.05
BerTSP-Approx	6.99	25.57	0.50	72.61	57.69
BerTSP-Ensemble	7.79	25.11	0.48	72.17	58.02

Table 3: Result for B-TSort and BerTSP on all 4 datasets. Both BerTSP-Approx and BerTSP-Ensemble outperform the baseline B-TSort for overall dataset results. (Results for BTSort are different from Prabhumoye et al. (2020) since we analyze the worst-case results considering all possible orderings for preprocessing (Section 4.3)).

We also note that both the variants of the proposed method outperform B-TSort. Among the two, BerTSP-Ensemble is consistently better as per PMR. For all the other metrics, both show competitive performance and BerTSP-Approx even beats BerTSP-Ensemble for some metrics/datasets. This shows that our simple heuristic approach works pretty well even though it is computationally inexpensive. This could be attributed to the fact that averages (or sums, equivalent in this case) tend to discount the bad predictions. They regress the overall representation of a sentence towards the good predictions (correctly predicted pair-orders) if they are in majority. This way we get a single-valued representation for a sentence that has information from both the local and global contexts.

5 Analysis and Discussions

In this section, we analyze our model on additional parameters including scalability, end-point performance, and displacement. Note that in this section, we use the BerTSP-Approx method for comparison

Dataset	Model	PMR	ACC	TAU	R-S	LCS
NIPS	**B-TSort**	0.00	28.30	0.59	78.67	63.52
	BerTSP	0.00	34.59	0.67	83.11	64.15
AAN	**B-TSort**	0.00	28.48	0.53	74.63	65.63
	BerTSP	0.00	29.41	0.55	76.06	64.09
NSF	**B-TSort**	0.04	17.29	0.41	70.96	56.43
	BerTSP	0.01	17.96	0.45	71.69	51.31

Table 4: Results for cases with more than 10 sentences.

Model	First	Last	First	Last
Dataset	**SIND**		**NIPS**	
B-TSort	77.19	57.03	89.55	74.38
BerTSP	78.85	59.33	92.54	74.63
Dataset	**AAN**		**NSF**	
B-TSort	88.76	78.78	61.13	40.55
BerTSP	90.10	79.55	66.95	47.13

Table 5: Prediction accuracy for first and last sentences.

with B-TSort as the BerTSP-Approx variant can be uniformly applied across all the datasets under consideration, as opposed to BerTSP-Ensemble which is a mixture of two methods.

5.1 Longer Sequences

To compare the scalability of models, we present the results for samples with more than 10 sentences. NIPS, AAN, and NSF qualify for this analysis while SIND doesn't as all samples in SIND have exactly 5 sentences. Table 4 shows the results on all metrics. BerTSP is dominant over B-TSort. For NIPS, we obtain a 22% improvement in position-wise sentence accuracy and 14% in Kendall Tau as compared to 6% and 3% respectively for all samples showing that the relative improvement is more for longer sequences.

5.2 First and Last Sentences

Table 5 shows the accuracy of prediction of first and last sentences of the sequence. This analysis is important as the end-points are crucial positions of the sequence and the first prediction is often decisive to the prediction of the rest of the sequence. BerTSP clearly overtakes B-TSort across all the datasets. For the NSF dataset which primarily has longer sequences (mean length $\sim$ 9), it achieves improvements of 10% and 16% on the prediction of first and last sentences respectively showing its efficacy for longer sequences.

5.3 Sentence Displacement

We perform sentence displacement analysis where we find the percentage of sentences for which the predicted position is within a window W (forward or backward) of its position in the correct order. For instance, if the correct position is 5 and $W = 1$, the predicted position should be within 4 to 6 to be included in the percentage. Naturally, a larger window allows for more displacement and hence the percentage is higher compared to smaller windows. Table 6 shows the results for $W = 1, 2, 3$. BerTSP clearly outperforms B-TSort across all datasets and window sizes. The improvement is generally more pronounced for smaller window sizes. Note that this metric is essentially a generalization of position-wise Sentence Accuracy for which $W = 0$.

5.4 Qualitative Examples

We also present three examples from the SIND captions dataset where BerTSP improves on B-TSort. The predicted orders for each example by B-TSort and BerTSP are also shown (Table 7).

6 Related Work

In the recent past, numerous neural approaches have been proposed for Sentence Ordering. Chen et al. (2016) used a pairwise ranking model (Zheng et al., 2007) that assigns a score to the relative ordering of every pair of sentences. Prabhumoye et al. (2020) train a classifier to predict an order constraint between any two sentences and use sorting based on these constraints to predict the final order. Zhu et al. (2021) construct multiple constraint graphs which are integrated into sentence representations by Graph Isomorphism Networks and ranked via ListMLE. The use of a pointer decoder for sequential prediction (Gong et al., 2016; Logeswaran et al., 2018; Wang and Wan, 2019; Oh et al., 2019) along with an intermediate paragraph encoder (Cui et al., 2018; Yin et al., 2019, 2020; Cui et al., 2020) for better capturing the global dependencies has been proposed in many variants. Kumar et al. (2020) replace the pointer decoder with a feed-forward neural network and use ranking loss to enable simultaneous prediction of scores for all sentences.

The pairwise approaches generally suffer from lack of global interactions while pointer-based ap-

Dataset	SIND			NIPS			AAN			NSF		
Model	**W=1**	**W=2**	**W=3**	**W=1**	**W=2**	**W=3**	**W=1**	**W=2**	**W=3**	**W=1**	**W=2**	**W=3**
B-TSort	79.44	93.20	98.69	85.19	93.35	97.02	87.64	92.82	97.77	49.29	64.55	74.59
BerTSP	81.28	94.37	99.11	86.08	94.35	97.72	88.81	95.78	98.19	49.57	64.62	74.48

Table 6: Results of displacement analysis of sentences on all datasets (W=Window size)

Example 1
s_1: We went to the park and a deer followed us.
s_2: We pet it and took pictures with it.
s_3: Then we traveled to the marvelous hotel.
s_4: The grounds were so immaculate.
s_5: We also got great pictures of the outside decor.
B-TSort: $s_2 \rightarrow s_1 \rightarrow s_3 \rightarrow s_5 \rightarrow s_4$
BerTSP: $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5$
Example 2
s_1: I went on vacation last year.
s_2: It was a beautiful place.
s_3: There were a lot of flower stores.
s_4: The buildings were very old.
s_5: There were a lot of other tourists there too.
B-TSort: $s_1 \rightarrow s_3 \rightarrow s_2 \rightarrow s_5 \rightarrow s_4$
BerTSP: $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_5 \rightarrow s_4$
Example 3
s_1: There were many people at the protest.
s_2: They had many signs.
s_3: And flags as well.
s_4: They did not like the war.
s_5: And made this known before leaving.
B-TSort: $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_5 \rightarrow s_4$
BerTSP: $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5$

Table 7: Qualitative comparison of B-TSort & BerTSP

proaches lag in utilizing the local pairwise context. The more recent works try to overcome this by incorporating the local relative ordering information into the pointer decoder (Yin et al., 2020; Cui et al., 2020). Also, likelihood-based decoding is prone to degeneration (Holtzman et al., 2020) especially for longer sequences of sentences and paragraph encoders can only capture limited information from every sentence. Lastly, the large memory requirement is the underlying issue with common neural approaches. In particular, (Cui et al., 2020) show better results compared to our approach (BerTSP) but methods such as BerTSP are much more memory efficient (Prabhumoye et al., 2020).

We have proposed a Traveling Salesman Problem based formulation for sentence ordering. The use of such graph-based optimizations have been explored in past work in NLP such as ARM-to-text generation (Song et al., 2016), opinion summarization (Nishikawa et al., 2010), multi-document summarization (Al-Saleh and Menai, 2018), etc.

7 Conclusion

We demonstrate the potential of a simpler, cheaper yet effective approach for the task of Sentence Ordering. Our reformulation of the task as an asymmetric TSP allows for the application of exact and heuristic graphical algorithms which are lighter and more transparent as opposed to heavier neural approaches.

Future Work: In place of BERT, use of alternative model architectures like Albert, XLNet and BART may provide better sentence representations for sentence ordering as their language modeling objectives better align with this task. In the decoding part, more heuristic-based or neural-based approaches to combinatorial optimization may provide better alternatives.

Acknowledgments

We thank anonymous INLG reviewers for providing valuable feedback.

Ethics Statement

We do not see any major ethical concerns arising out of our work. Our work focuses only on finding the coherent ordering of sentences.

References

Asma Al-Saleh and Mohamed El Bachir Menai. 2018. Solving multi-document summarization as an orienteering problem. *Algorithms*, 11(7):96.

Evelin Amorim, Marcia Cançado, and Adriano Veloso. 2018. Automated essay scoring in the presence of biased ratings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 229–237.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment*, 4(3).

Regina Barzilay and Noemie Elhadad. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17:35–55.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Neural sentence ordering. *arXiv preprint arXiv:1607.06952*.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2018. Deep attentive sentence ordering network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4340–4349, Brussels, Belgium. Association for Computational Linguistics.

Baiyun Cui, Yingming Li, and Zhongfei Zhang. 2020. BERT-enhanced relational sentence ordering network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6310–6320, Online. Association for Computational Linguistics.

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

Youmna Farag, Helen Yannakoudakis, and Ted Briscoe. 2018. Neural automated essay scoring and coherence modeling for adversarially crafted input. *arXiv preprint arXiv:1804.06898*.

Varun Gangal, Steven Y. Feng, Eduard H. Hovy, and Teruko Mitamura. 2021. NAREOR: the narrative reordering problem. *CoRR*, abs/2104.06669.

Jingjing Gong, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. End-to-end neural sentence ordering using pointer network. *arXiv preprint arXiv:1611.04953*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, and Dhruv Batra. et almbox. 2016. visual storytelling. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2020. Narrative text generation with a latent discrete plan. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3637–3650.

Harsh Jhamtani and Peter Clark. 2020. Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 137–150.

Pawan Kumar, Dhanajit Brahma, Harish Karnick, and Piyush Rai. 2020. Deep attentive ranking networks for learning to order sentences. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8115–8122.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 545–552.

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*.

Hitoshi Nishikawa, Takaaki Hasegawa, Yoshihiro Matsuo, and Genichiro Kikui. 2010. Opinion summarization with integer linear programming formulation for sentence extraction and ordering. In *Coling 2010: Posters*, pages 910–918.

Byungkook Oh, Seungmin Seo, Cheolheon Shin, Eunju Jo, and Kyong-Ho Lee. 2019. Topic-guided coherence modeling for sentence ordering by preserving global and local information. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2273–2283, Hong Kong, China. Association for Computational Linguistics.

Shrimai Prabhumoye, Ruslan Salakhutdinov, and Alan W Black. 2020. Topological sort for sentence ordering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. proscript: Partially ordered scripts generation via pre-trained language models. *CoRR*, abs/2104.08251.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D. Manning. 2019. Do massively pretrained language models make better storytellers?

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. Amr-to-text generation as a traveling salesman problem. *arXiv preprint arXiv:1609.07451*.

Tianming Wang and Xiaojun Wan. 2019. Hierarchical attention networks for sentence ordering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7184–7191.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yongjing Yin, Fandong Meng, Jinsong Su, Yubin Ge, Lingeng Song, Jie Zhou, and Jiebo Luo. 2020. Enhancing pointer network for sentence ordering with pairwise ordering predictions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9482–9489.

Yongjing Yin, Linfeng Song, Jinsong Su, Jiali Zeng, Chulun Zhou, and Jiebo Luo. 2019. Graph-based neural sentence ordering. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5387–5393. International Joint Conferences on Artificial Intelligence Organization.

Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. 2007. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 287–294.

Yutao Zhu, Kun Zhou, Jian-Yun Nie, Shengchao Liu, and Zhicheng Dou. 2021. Neural sentence ordering based on constraint graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14656–14664.

A Additional Method Details

A.1 Dummy Node for ATSP:

TSP requires the source to be pre-specified. Since it finds a cycle, the choice of the source s is irrelevant. But for our purpose, we need to find an order which is inherently not cyclic. To deal with this, we introduce an additional node in the graph, v_{n+1}, which is at 0 distance from all other vertices in either direction, i.e. $w_{(n+1)i} = w_{i(n+1)} = 0, \forall\ i$. This dummy node serves as the source from which the cycle begins and consequently terminates. Since it is at 0 distance from all nodes, its addition does not affect the minimization process.

A.2 Heuristic solution for TSP:

We propose a heuristic solution that is cheap and accounts for the order along with the cost. Consider the unaugmented weight matrix: $W = \{w_{ij}\}_{\forall i,j \in \{1,2,...,n\}}$ where $w_{ij} = 1 - P(s_i \rightarrow s_j)$. Consider the following sum of probabilities:

$$P(s_i \rightarrow s_2) + P(s_i \rightarrow s_3) + ... + P(s_i \rightarrow s_n)$$

If this sum is high, s_i has a high probability of occurring before $\{1, 2, ..., i-1, i+1, ..., n\}$ in the predicted sequence. Consequently, if we take the sum of complement probabilities, the lower the sum, the higher is the probability of s_i occurring before $\{1, 2, ..., i-1, i+1, ..., n\}$ in the predicted sequence. This complementary sum is nothing but the row-sum of row i of matrix W:

$$w_{i1} + w_{i1} + ... + w_{in}\ (w_{ii} = 0)$$

Hence, we take the row-sums of all rows in W and sort them in ascending order. This gives us the predicted order. Since it involves simple sorting, its runtime complexity is $O(nlogn)$.

B Implementation details

The code for the sentence-pair encoder and the evaluation metrics is derived from (Prabhumoye et al., 2020)[2]. The hyperparameters values are also taken from this work. The experiments are conducted on GeForce RTX 2080 Ti GPU. Our code[3] is written using Pytorch deep learning framework.

[2] https://github.com/shrimai/Topological-Sort-for-Sentence-Ordering

[3] https://github.com/vkeswani/BerTSP

Dataset	Mean Length	Train	Dev	Test
SIND	5	40155	4990	5055
NIPS	6	2448	409	402
AAN	5	8569	962	2626
NSF	9	96070	10185	21580

Table 8: Descriptive statistics of the four datasets considered

C Data

C.1 Access

The SIND captions dataset is available online[4]. Note that the Stories of Images-in-Sequence (SIS) portion is the one relevant to our task. The abstract datasets, NIPS, AAN, and NSF, were obtained from Logeswaran et al. (2018).

C.1.1 Descriptive Statistics

We present the mean length and the split of the four datasets into train, development, and test sets in table 8.

[4] https://visionandlanguage.net/VIST/dataset.html

Underreporting of errors in NLG output, and what to do about it

Emiel van Miltenburg,[1*] **Miruna Clinciu,**[2,3,4] **Ondřej Dušek,**[5] **Dimitra Gkatzia,**[6]
Stephanie Inglis,[7] **Leo Leppänen,**[8] **Saad Mahamood,**[9] **Emma Manning,**[10]
Stephanie Schoch,[11] **Craig Thomson,**[7] and **Luou Wen**[12]

[1]Tilburg University, [2]Edinburgh Centre for Robotics, [3]Heriot-Watt University,
[4]University of Edinburgh, [5]Charles University, Prague, [6]Edinburgh Napier University,
[7]University of Aberdeen, [8]University of Helsinki, [9]trivago N.V., [10]Georgetown University,
[11]University of Virginia, [12]University of Cambridge
Contact: c.w.j.vanmiltenburg@tilburguniversity.edu

Abstract

We observe a severe under-reporting of the different kinds of errors that Natural Language Generation systems make. This is a problem, because mistakes are an important indicator of where systems should still be improved. If authors only report overall performance metrics, the research community is left in the dark about the specific weaknesses that are exhibited by 'state-of-the-art' research. Next to quantifying the extent of error under-reporting, this position paper provides recommendations for error identification, analysis and reporting.

1 Introduction

This paper turned out very differently from the one we had initially intended to write. Our original intention was to write an overview of the different kinds of errors that appear in the output of different kinds of Natural Language Generation (NLG) systems, and to develop a general taxonomy of NLG output errors, based on the publications that have appeared at previous INLG conferences (similar to Howcroft et al. 2020; Belz et al. 2020). This, however, turned out to be impossible. The reason? There is a severe under-reporting of the different kinds of errors that NLG systems make. By this assertion, we mean that authors neither include any error analysis nor provide any examples of errors made by the system, and they do not make reference to different kinds of errors that may appear in the output. The latter is a lower bar than carrying out an error analysis, which requires a more systematic approach where several outputs are sampled and analysed for the presence of errors, which are then categorised (ideally through a formal procedure with multiple annotators). Section 3 provides more detailed statistics about error reporting in different years of INLG (and ENLG), and the amount of papers that discuss the kinds of errors that may appear in NLG output.

The fact that errors are under-reported in the NLG literature is probably unsurprising to experienced researchers in this area. The lack of reporting of negative results in AI has been a well-known issue for many years (Reiter et al., 2003). With the classic NLG example being the reporting of negative results for the STOP project on smoking cessation (Reiter et al., 2001, 2003). But even going in with (relatively) low expectations, it was confronting just to see how little we as a community look at the mistakes that our systems make.

We believe that it is both necessary *and* possible to improve our ways. One of the reasons why it is necessary to provide more error analyses (see §2.2 for more), is that otherwise, it is unclear what are the strengths and weaknesses of current NLG systems. In what follows, we provide guidance on how to gain more insight into system behavior.

This paper provides a general framework to carry out error analyses. First we cover the terminology and related literature (§2), after which we quantify the problem of under-reporting (§3). Following up on this, we provide recommendations on how to carry out an error analysis (§4). We acknowledge that there are barriers to a more widespread adoption of error analyses, and discuss some ways to overcome them (§5). Our code and data are provided as supplementary materials.

2 Background: NLG systems and errors

2.1 Defining errors

There are many ways in which a given NLG system can fail. Therefore it can be difficult to exactly define all the different types of errors that can possibly occur. Whilst error analyses in past NLG literature were not sufficient for us to create a taxonomy, we will instead propose high-level distinctions to help

*This project was led by the first author. Remaining authors are presented in alphabetical order.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 140–153,
Aberdeen, Scotland, UK, 20-24 September 2021.

bring clarity within the NLG research community.

This paper focuses on text errors, which we define as countable instances of things that went wrong, as identified from the generated text.[1] Text errors apply when something is incorrect in the generated text with respect to the data, an external knowledge source, or the communicative goal.

Through our focus on text errors, we only look at the *product* (what comes out) of an NLG system, so that we can compare the result of different kinds of systems (e.g., rule-based pipelines versus neural end-to-end systems), with error categories that are independent of the *process* (how the text is produced).[2] For completeness, we discuss errors related to the production process in §2.3.

By *error analysis* we mean the identification and categorisation of errors, after which statistics about the distribution of error categories are reported. It is an annotation process (Pustejovsky and Stubbs, 2012; Ide and Pustejovsky, 2017), similar to Quantitative Content Analysis in the social sciences (Krippendorff, 2018; Neuendorf, 2017).[3] Error analysis can be carried out during development (to see what kinds of mistakes the system is currently making), as the last part of a study (evaluating a new system that you are presenting), or as a standalone study (comparing different systems). The latter option requires output data to be available, ideally for both the validation and test sets. A rich source of output data is the GEM shared task (Gehrmann et al., 2021).

Text errors can be categorised in several different types, including factual errors (e.g. incorrect number; Thomson and Reiter 2020), and errors related to form (spelling, grammaticality), style (formal versus informal, empathetic versus neutral), or behavior (over- and under-specification). Some of these are universally wrong, while others may be 'contextually wrong' with respect to the task success or for a particular design goal. For example, formal texts aren't wrong *per se*, but if the goal is to produce informal texts, then any semblance of formality may be considered incorrect.

It may be possible to relate different kinds of errors to the different dimensions of text quality identified by Belz et al. (2020). What is crucial here, is that we are able to identify the specific thing which went wrong, rather than just generate a number that is representative of overall quality.

2.2 Why do authors need to report errors?

There is a need for realism in the NLG community. By providing examples of different kinds of errors, we can show the complexity of the task(s) at hand, and the challenges that still lie ahead. This also helps set realistic expectations for users of NLG technology, and people who might otherwise build on top of our work. A similar argument has been put forward by Mitchell et al. (2019), arguing for 'model cards' that provide, inter alia, performance metrics based on quantitative evaluation methods. We encourage authors to also look at the data and provide examples of where systems produce errors. Under-reporting the types of errors that a system makes is harmful because it leaves us unable to fully appreciate the system's performance.

While some errors may be detected automatically, e.g., using information extraction techniques (Wiseman et al., 2017) or manually defined rules (Dušek et al., 2018), others are harder or impossible to identify if not reported. We rely on researchers to communicate the less obvious errors to the reader, to avoid them going unnoticed and causing harm for subsequent users of the technology.

Reporting errors is also useful when comparing different implementation paradigms, such as pipeline-based data-to-text systems versus neural end-to-end systems. It is important to ask where systems fall short, because different systems may have different shortcomings. One example of this is the E2E challenge, where systems with similar human rating scores show very different behavior (Dušek et al., 2020).

Finally, human and automatic evaluation metrics, or at least the ones that generate some kind of intrinsic rating, are too coarse-grained to capture relevant information. They are general evaluations of system performance that estimate an average-case performance across a limited set of abstract dimensions (if they measure anything meaningful

[1] We use the term 'text' to refer to any expression of natural language. For example, sign language (as in Mazzei 2015) would be considered 'text' under this definition.

[2] By focusing on countable instances of things that went wrong in the output text, we also exclude issues such as bias and low output diversity, that are global properties of the collection of outputs that a system produces for a given amount of inputs, rather than being identifiable in individual outputs.

[3] There has been some effort to automate this process. For example, Shimorina et al. (2021) describe an automatic error analysis procedure for shallow surface realisation, and Stevens-Guille et al. (2020) automate the detection of repetitions, omissions, and hallucinations. However, for many NLG tasks, this kind of automation is still out of reach, given the wide range of possible correct outputs that are available in language generation tasks.

at all; see Reiter 2018). We don't usually know the worst-case performance, and we don't know what kinds of errors cause the metrics or ratings to be sub-optimal. Additionally, the general lack of extrinsic evaluations among NLG researchers (Gkatzia and Mahamood, 2015) means that in some cases we only have a partial understanding of the possible errors for a given system.

2.3 Levels of analysis

As noted above, our focus on errors in the output text is essential to facilitate framework-neutral comparisons between the performance of different systems. When categorizing the errors made by different systems, it is important to be careful with terms such as *hallucination* and *omission*, since these are process-level (pertaining to the system) rather than product-level (pertaining to the output) descriptions of the errors.[4] Process-level descriptions are problematic because we cannot reliably determine how an error came about, based on the output alone.[5] We can distinguish between at least two causes of errors, which we define below: system problems and data problems. While these problems should be dealt with, we do not consider them to be the subject of error analysis.

System problems can be defined as the malfunctioning of one or several components in a given system, or the malfunctioning of the system as a whole. System problems in rule/template-based systems could be considered as synonymous to 'bugs,' which are either semantic and/or syntactic in nature. If the system has operated in a mode other than intended (e.g., as spotted through an error analysis), the problem has to be identified, and then corrected. Identifying and solving such problems may require close involvement of domain experts for systems that incorporate significant domain knowledge or expertise (Mahamood and Reiter, 2012). Van Deemter and Reiter (2018) provide further discussion of how errors could occur at different stages of the NLG pipeline system. System problems in end-to-end systems are harder to identify, but recent work on interpretability/explainability aims to improve this (Gilpin et al., 2019).

Data problems are inaccuracies in the input that are reflected in the output. For example: when a player scored three goals in a real-world sports game, but only one goal is recorded (for whatever reason) in the data, even a perfect NLG system will generate an error in its summary of the match. Such errors may be identified as factual errors by cross-referencing the input data with external sources. They can then be further diagnosed as data errors by tracing back the errors to the data source.

3 Under-reporting of errors

We examined different *NLG conferences to determine the amount of papers that describe (types of) output errors, and the amount of papers that actually provide a manual error analysis.

3.1 Approach

We selected all the papers from three SIGGEN conferences, five years apart from each other: INLG2010, ENLG2015, and INLG2020. We split up the papers such that all authors looked at a selection of papers from one of these conferences, and informally marked all papers that discuss NLG errors in some way. These papers helped us define the terms 'error' and 'error analysis' more precisely.

In a second round of annotation, multiple annotators categorised all papers as 'amenable' or 'not amenable' to an error analysis. A paper is amenable to an error analysis if one of its primary contributions is presenting an NLG system that produces some form of output text. So, NLG experiments are amenable to an error analysis, while survey papers are not.[6] For all amenable papers, the annotator indicated whether the paper (a) mentions any errors in the output and (b) whether it contains an error analysis.[7] We encouraged discussion between annotators whenever they felt uncertain (details in Appendix A). The annotations for each paper were subsequently checked by one other annotator, after which any disagreements were adjudicated through

[4] Furthermore, terms like *hallucination* may be seen as unnecessary anthropomorphisms that trivialise mental illness.

[5] A further reason to avoid process-level descriptors is that they are often strongly associated with one type of approach. For example, the term 'hallucination' is almost exclusively used with end-to-end systems, as it is common for these systems to add phrases in the output text that are not grounded in the input. In our experience, pipeline systems are hardly ever referred to as 'hallucinating.' As such, it is better to avoid the term and instead talk about concrete phenomena in the output.

[6] Examples of other kinds of papers that are not amenable include evaluation papers, shared task proposals, papers which analyze patterns in human-produced language, and papers which describe a component in ongoing NLG work which does not yet produce textual output (e.g. a ranking module).

[7] As defined in § 2, errors are (countable) instances of something that is wrong about the output. An 'error mention' is a reference to such an instance or a class of such instances. Error analyses are formalised procedures through which annotators identify and categorise errors in the output.

Venue	Total	Amenable	Error mention	Error analysis	Percentage with error analysis
INLG2010	37	16	6	0	0%
ENLG2015	28	20	4	1	5%
INLG2020	46	35	19	4	11%

Table 1: Annotation results for different SIGGEN conferences, showing the percentage of amenable papers that included error analyses. Upon further inspection, most error mentions are relatively general/superficial.

a group discussion.

3.2 Results

Table 1 provides an overview of our results. We found that only five papers at the selected *NLG conferences provide an error analysis,[8] and more than half of the papers fail to mention any errors in the output. This means that the INLG community is systematically under-informed about the weaknesses of existing approaches. In light of our original goal, it does not seem to be a fruitful exercise to survey all SIGGEN papers if so few authors discuss any output errors. Instead, we need a culture change where authors discuss the output of their systems in more detail. Once this practice is more common, we can start to make generalisations about the different kinds of errors that NLG systems make. To facilitate this culture change, we give a set of recommendations for error analysis.

4 Recommendations for error analysis

We provide general recommendations for carrying out an error analysis, summarized in Figure 1.

4.1 Setting expectations

Before starting, it is important to be clear about your goals and expectations for the study.

Goal Generally speaking, the goal of an error analysis is to find and quantify system errors statistically, to allow a thorough comparison of different systems, and to help the reader understand the shortcomings of your system. But your personal goals and interests may differ. For example, you may only be interested in *grammatical* errors, and less so in factual errors.

Expected errors When starting an error analysis, you may already have some ideas about what kinds of errors might appear in the outputs of different systems. These ideas may stem from the literature (theoretical limitations, or discussions of errors), from your personal experience as an NLG researcher, or it might just be an impression you have from talking to others. You might also have particular expectations about what the distribution of errors will look like.

Both goals and expectations may bias your study, and cause you to overlook particular kinds of errors. But if you are aware of these biases, you may be able to take them into account, and later check if the results confirm your original expectations. Hence, it may be useful to preregister your study, so as to make your thoughts and plans explicit (Haven and Grootel, 2019; van Miltenburg et al., 2021). This also makes it easier for others to check whether they agree with the assumptions behind your study.

4.2 Set-up

Given your goals and expectations, there are several design choices that you have to make, in order to carry out your study.

Systems and outputs Since error analysis is relatively labor-intensive, it may not be feasible to look at a wide array of different systems. In that case, you could pre-select a smaller number of models, either based on automatic metric scores, or based on specific model features you are interested in. Alternatively, you could see to what extent the model outputs overlap, given the same input. If two models produce exactly the same output, you only need to annotate that output once.

Number of outputs Ideally, the number of outputs should be based on a power analysis to provide meaningful comparisons (Card et al., 2020; van der Lee et al., 2021), but other considerations, such as time and budget, may be taken into account.

Sample selection Regarding the selection of examples to analyze, there are three basic alternatives: The most basic is *random sampling* from the validation/test outputs. Another option is selecting *specific kinds of inputs* and analysing all corresponding outputs. Here, inputs known to be difficult/adversarial or inputs specifically targeting system properties or features may be selected

[8] Summaries of these error analyses are in Appendix B.

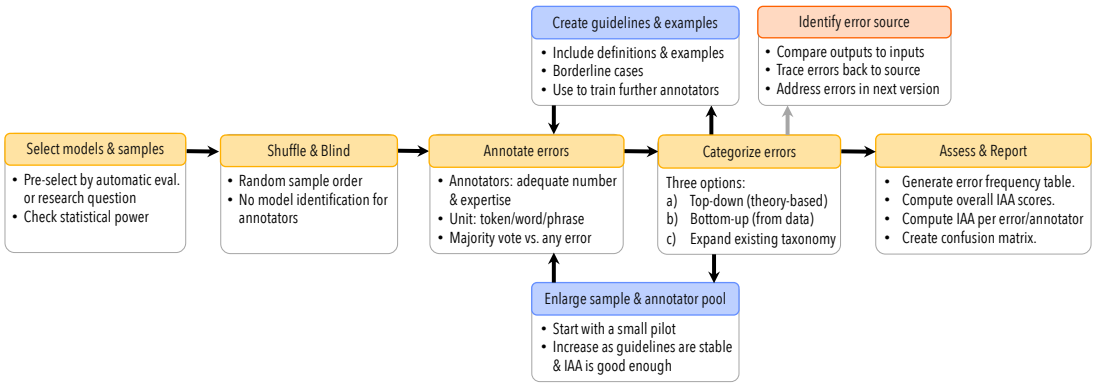


Figure 1: Flowchart depicting recommended analysis steps, as described in §4. IAA stands for Inter-Annotator Agreement, as measured through Cohen's kappa or Krippendorff's alpha, for example.

(Ribeiro et al., 2020). Finally, examples to analyze may also be selected based on *quantitative values*: automatic metric scores or ratings in a preceding general human evaluation. This way, error analysis can provide explanation for the automatic or human scores. The most suitable option depends on your specific use case: While random selection gives the least biased picture of the model performance, selecting specifically hard and/or low-rated samples may be more efficient. Also note that the sample selection should always be independent of any samples you may have previously examined during system development, since any errors for those cases are likely to have been resolved already (although you cannot be sure until you have verified these cases as well).

Presentation The order of the items should be randomized (to reduce possible order effects), and if multiple system variants are considered, the annotators must not know which system produced which output (to minimise annotator bias).

Interface The efficiency of any annotation task depends on the quality of the interface. With the right interface, annotators may be able to annotate more data in a shorter time frame. Monarch (2021, Chapter 11) provides recommendations on interface design based on principles from the field of Human-Computer Interaction (HCI). Once you have a working interface, it is important to test the interface and obtain feedback from annotators to see whether it can be made more intuitive or efficient (e.g. by adding keyboard shortcuts to perform common operations).[9]

Annotators and the annotation process The annotation process can be split into two parts: identifying the errors (§4.3), and categorising the errors (§4.4). These can either be carried out sequentially (first identify, then categorize) or simultaneously (asking annotators to both identify and categorize errors at the same time). The choices you make here also impact annotator requirements, and the evaluation of the annotation procedure.

Number of annotators Generally speaking, having more annotators reduces the prevalence of the individual bias (Artstein and Poesio, 2005). This is particularly relevant if we want to detect all the errors in the output data. Having more annotators means that we are less likely to overlook individual instances of errors. Once those errors are identified, it may make more sense to rely on a smaller set of well-trained annotators to categorise the different errors. In the ideal situation, all errors are annotated by (at least) two judges so as to be able to detect and resolve any disagreements afterwards. If this is not possible, then you should at least double-annotate a large enough sample to reliably estimate inter-annotator agreement.[10]

Role of the annotators Ideally, the annotators should be independent of the authors reporting the error analysis (Neuendorf, 2017), to ensure that the results are not influenced by any personal biases about the systems involved, and that the annotations are indeed based on the guidelines themselves rather than on discussions between the authors. If this is not feasible, then the authors should at least ensure that they remain ignorant of the identity of

[9] Note that keyboard operations are generally quicker than using the mouse (Monarch, 2021).

[10] See Krippendorff 2011 for a reference table to determine the sample size for Krippendorff's α. Similar studies exist for Cohen's κ, e.g. Flack et al. 1988; Sim and Wright 2005.

the system that produced the relevant outputs.

Degree of expertise Depending on the complexity of the annotation guidelines, the error analysis may require expertise in linguistics (in the case of a theory-driven error categorisation scheme), or the relevant application area (with a context-driven error categorisation scheme). For example, Mahamood and Reiter (2012) worked with nurses to identify errors in reports generated for parents of neonatal infants. Taking into consideration the costly process of selecting domain expert annotators and the importance of quality control, non-domain experts might be also considered, ensuring their qualification through (intensive) training (Artstein and Poesio, 2005; Carlson et al., 2001).[11]

Compensation and treatment of workers If annotators are hired, either directly or via a crowdsourcing platform such as MTurk, they should be compensated and treated fairly (Fort et al., 2011). Silberman et al. (2018) provide useful guidelines for the treatment of crowd-workers. The authors note that they should at least be paid the minimum wage, they should be paid promptly, and they should be treated with respect. This means you should be ready to answer questions about the annotation task, and to streamline the task based on worker feedback. If you use human participants to annotate the data, you likely also need to apply for approval by an Institutional Review Board (IRB).

Training Annotators should receive training to be able to carry out the error analysis, but the amount of training depends on the difficulty of the task (which depends, among other factors, on the *coding units* (see §4.3), and the number of error types to distinguish). They should be provided with the annotation guidelines (§4.5), and then be asked to annotate texts where the errors are known (but not visible). The solutions would ideally be created by experts, although in some cases, solutions created by researchers may be sufficient (Thomson and Reiter, 2020). It should be decided in advance what the threshold is to accept annotators for the reaming work, and, if they fail, whether to provide additional training or find other candidates. Note that annotators should also be compensated for taking part in the training (see previous paragraph).

[11] At least on the MTurk platform, Requesters can set the entrance requirements for their tasks such that only Workers who passed a qualifying test may carry out annotation tasks.

4.3 Identifying the errors

Error identification focuses on discovering all errors in the chosen output samples (as defined in the introduction). Previously, Popović (2020) asked error annotators to identify issues with comprehensibility and adequacy in machine-translated text. Similarly, Freitag et al. (2021) proposed a manual error annotation task where the annotators identified and highlighted errors within each segment in a document, taking into account the document's context as well as the severity of the errors.

The major challenge in this annotation step is how to determine the units of analysis; should annotators mark individual tokens, phrases, or constituents as being incorrect, or can they just freely highlight any sequence of words? In content analysis, this is called *unitizing*, and having an agreed-upon unit of analysis makes it easier to process the annotations and compute inter-annotator agreement (Krippendorff et al., 2016).[12] What is the right unit may depend on the task at hand, and as such is beyond the scope of this paper.[13]

A final question is what to do when there is disagreement between annotators about what counts as an error or not. When working with multiple annotators, it may be possible to use majority voting, but one might also be inclusive and keep all the identified errors for further annotation. The error categorization phase may then include a category for those instances that are not errors after all.

4.4 Categorizing errors

There are three ways to develop an error categorisation system:

1. **Top-down** approaches use existing theory to derive different types of errors. For example, Higashinaka et al. (2015a) develop an error taxonomy based on Grice's (1975) Maxims of conversation. And the top levels of Costa et al.'s (2015) error taxonomy[14] are based on general linguistic theory, inspired by Dulay et al. (1982).

2. **Bottom-up** approaches first identify different

[12] Though note that Krippendorff et al. do provide a metric to compute inter-annotator agreement for annotators who use units of different lengths.

[13] One interesting solution to the problem of unitization is provided by Pagnoni et al. (2021), who do not identify individual errors, but do allow annotators to "check all types that apply" at the sentence level. The downside of this approach is that it is not fine-grained enough to be able to count individual instances of errors, but you do get an overall impression of the error distribution based on the sentence count for each type.

[14] Orthography, Lexis, Grammar, Semantic, and Discourse.

errors in the output, and then try to develop coherent categories of errors based on the different kinds of attested errors. An example of this is provided by Higashinaka et al. (2015b), who use a clustering algorithm to automatically group errors based on comments from the annotators (verbal descriptions of the nature of the mistakes that were made). Of course, you do not have to use a clustering algorithm. You can also manually sort the errors into different groups (either digitally[15] or physically[16]).

3. **Expanding on existing taxonomies**: here we make use of other researchers' efforts to categorize different kinds of errors, by adding, removing, or merging different categories. For example, Costa et al. (2015) describe how different taxonomies of errors in Machine Translation build on each other. In NLG, if you are working on data-to-text, then you could take Thomson and Reiter's (2020) taxonomy as a starting point. Alternatively, Dou et al. (2021) present a crowd-sourced error annotation schema called SCARECROW. For image captioning, there is a more specific taxonomy provided by van Miltenburg and Elliott (2017). Future work may also investigate the possibility of merging all of these taxonomies and relating the categories to the quality criteria identified by Belz et al. (2020).

The problem of error ambiguity To be able to categorize different kinds of errors, we often rely on the *edit-distance heuristic*. That is: we say that the text contains particular kinds of errors, because fixing those errors will give us the desired output. With this reasoning, we take the mental 'shortest path' towards the closest correct text.[17] This at least gives us a set of 'perceived errors' in the text, that provides a useful starting point for future research. However, during the process of identifying errors, we may find that there are multiple 'shortest paths' that lead to a correct utterance, resulting in error ambiguity (see, e.g., Van Miltenburg and Elliott 2017; Thomson and Reiter 2020, §3.3).

For example, if the output text from a sports summary system notes that Player A scored 2 points, while in fact Player A scored 1 point and Player B scored 2 points, should we say that this is a number error (2 instead of 1) or a person error (Player A instead of B)? This example also shows the fragility of the distinction between product and process. It is very tempting to look at what the system did to determine the right category, but it is unclear whether the 'true error category' is always knowable.

There are multiple ways to address the problem of error ambiguity. For instance, we may award partial credit ($1/n$ error categories), mark both types of errors as applying in this situation (overgeneralising, to be on the safe side), or count all ambiguous cases to separately report on them in the overall frequency table. Another solution, used by Thomson and Reiter (2020) is to provide the annotators with a fixed preference order (NAME, NUMBER, WORD, CONTEXT), so that similar cases are resolved in a similar fashion.

4.5 Writing annotation guidelines

Once you have determined an error identification strategy and developed an error categorisation system, you should describe these in a clear set of annotation guidelines. At the very least, these guidelines should contain relevant definitions (of each error category, and of errors in general), along with a set of examples, so that annotators can easily recognize different types of errors. For clarity, you may wish to add examples of borderline cases with an explanation of why they should be categorized in a particular way.

Pilot The development of a categorisation system and matching guidelines is an iterative process. This means that you will need to carry out multiple pilot studies in order to end up with a reliable set of guidelines,[18] that is easily understood by the annotators, and provides full coverage of the data. Pilot studies are also important to determine how long the annotation will take. This is not just practical to plan your study, but also essential to determine how much crowd-workers should be paid per task, so that you are able to guarantee a minimum wage.

4.6 Assessment

Annotators and annotations can be assessed during or after the error analysis.[19]

[15] E.g. via a program like Excel, MaxQDA or Atlas.ti, or a website like https://www.well-sorted.org.

[16] A good example of this *pile sorting* method is provided by Yeh et al. (2014). Blanchard and Banerji (2016) give further recommendations.

[17] Note that we don't know whether the errors we identified are actually the ones that the system internally made. This would require further investigation, tracing back the origins of each different instance of an error.

[18] As determined by an inter-annotator agreement that exceeds a particular threshold, e.g. Krippendorff's $\alpha \geq 0.8$.

[19] And in many cases, the annotators will already have been assessed during the training phase, using the same measures.

During the error analysis Particularly with crowd-sourced annotations it is common to include gold-standard items in the annotation task, so that it is possible to flag annotators who provide too many incorrect responses. It is also possible to carry out an intermediate assessment of inter-annotator agreement (IAA), described in more detail below. This is particularly relevant for larger projects, where annotators may diverge over time.

After the error analysis You can compute IAA scores (e.g., Cohen's κ or Krippendorff's α, see: Cohen 1960; Krippendorff 1970, 2018), to show the overall reliability of the annotations, the pairwise agreement between different annotators, and the reliability of the annotations for each error type. You can also produce a confusion matrix; a table that takes one of the annotators (or the adjudicated annotations after discussion) as a reference, and provides counts for how often errors from a particular category were annotated as belonging to any of the error categories (Pustejovsky and Stubbs, 2012). This shows all disagreements at a glance.

Any analysis of (dis)agreement or IAA scores requires there to be overlap between the annotators. This overlap should be large enough to reliably identify any issues with either the guidelines or the annotators. Low agreement between annotators may be addressed by having an adjudication round, where the annotators (or an expert judge) resolve any disagreements; rejecting the work of unreliable annotators; or revising the task or the annotation guidelines, followed by another annotation round (Pustejovsky and Stubbs, 2012).

4.7 Reporting

We recommend that authors should provide a table reporting the frequency of each error type, along with the relevant IAA scores. The main text should at least provide the overall IAA score, while IAA scores for the separate error categories could also be provided in the appendix. For completeness, it is also useful to include a confusion matrix, but this can also be put in the appendix. The main text should provide a discussion of both the frequency table, as well as the IAA scores. What might explain the distribution of errors? What do the examples from the *Other*-category look like? And how should we interpret the IAA score? Particularly with low IAA scores, it is reasonable to ask why the scores are so low, and how this could be improved. Reasons for low IAA scores include: unclear annotation guidelines, ambiguity in the data, and having one or more unreliable annotator(s). The final annotation guidelines should be provided as supplementary materials with your final report. All annotations and output data (e.g. train, validation, and test outputs, possibly with confidence scores) should of course also be shared.

5 (Overcoming) barriers to adoption

One reason why authors may feel hesitant about providing an error analysis is that it takes up significantly more space than the inclusion of some overall performance statistics. The current page limits in our field may be too tight to include an error analysis. Relegating error analyses to the appendix does not feel right, considering the amount of work that goes into providing such an analysis. Given the effort that goes into an error analysis, authors have to make trade-offs in their time spent doing research. If papers can easily get accepted without any error analysis, it is understandable that this additional step is often avoided. How can we encourage other NLG researchers to provide more error analyses, or even just examples of errors?

Improving our standards We should adopt reporting guidelines that stress the importance of error analysis in papers reporting NLG experiments. The NLP community is already adopting such guidelines to improve the reproducibility of published work (see Dodge et al.'s (2019) reproducibility checklist that authors for EMNLP2020 need to fill in). We should also stress the importance of error reporting in our reviewing forms; authors should be rewarded for providing insightful analyses of the outputs of their systems. One notable example here is COLING 2018, which explicitly asked about error analyses in their reviewing form for NLP engineering experiments, and had a 'Best Error Analysis' award.[20,21]

Making space for error analyses We should make space for error analyses. The page limit in *ACL conferences is already expanding to incorporate ethics statements, to describe the broader impact of our research. This suggests that we have reached the limits of what fits inside standard papers, and an expansion is warranted. An alternative is to publish more journal papers, where there is more space to fit an error analysis, but then we as

[20] https://coling2018.org/paper-types/

[21] http://coling2018.org/index.html%3Fp=1558.html

a community also need to encourage and increase our appreciation of journal submissions.

Spreading the word Finally, we should inform others about how to carry out a proper error analysis. If this is a problem of exposure, then we should have a conversation about the importance of error reporting. This paper is an attempt to get the conversation started.

6 Follow-up work

What should you do after you have carried out an error analysis? We identify three directions for follow-up studies.

Errors in inputs An additional step can be added during the identification of errors which focuses on observing the system inputs and their relation to the errors. Errors in the generated text may occur due to semantically noisy (Dušek et al., 2019) or incorrect system input (Clinciu et al., 2021); for instance, input data values might be inaccurate or the input might not be updated due to a recent change (e.g., new president). To pinpoint the source of the errors, we encourage authors to look at their input data jointly with the output, so that errors in inputs can be identified as such.

Building new evaluation sets Once you have identified different kinds of errors, you can try to trace the origin of those errors in your NLG model, or posit a hypothesis about what kinds of inputs cause the system to produce faulty output. But how can you tell whether the problem is really solved? Or how can you stimulate research in this direction? One solution, following McCoy et al. (2019), is to construct a new evaluation set based on the (suspected) properties of the errors you have identified. Future research, knowing the scope of the problem from your error analysis, can then use this benchmark to measure progress towards a solution.

Scales and types of errors Error types and human evaluation scales are closely related. For example, if there are different kinds of grammatical errors in a text, we expect human grammaticality ratings to go down as well. But the relation between errors and human ratings is not always as transparent as with grammaticality. Van Miltenburg et al. (2020) show that different kinds of semantic errors have a different impact on the perceived overall quality of image descriptions.[22] Future research should aim to explore the connection between the two in more detail, so that there is a clearer link between different kinds of errors and different quality criteria (Belz et al., 2020).

7 Conclusion

Having found that NLG papers tend to underreport errors, we have motivated why authors should carry out error analyses, and provided a guide on how to carry out such analyses. We hope that this paper paves the way for more in-depth discussions of errors in NLG output.

Acknowledgements

We would like to thank Emily Bender and the anonymous reviewers for their feedback. Dimitra Gkatzia's contribution was supported under the EPSRC projects CiViL (EP/T014598/1) and NLG for Low-resource Domains (EP/T024917/1). Miruna Clinciu's contribution is supported by the EPSRC Centre for Doctoral Training in Robotics and Autonomous Systems at Heriot-Watt University and the University of Edinburgh. Miruna Clinciu's PhD is funded by Schlumberger Cambridge Research Limited (EP/L016834/1, 2018-2021). Ondřej Dušek's contribution was supported by Charles University grant PRIMUS/19/SCI/10. Craig Thomson's work is supported under an EPSRC NPIF studentship grant (EP/R512412/1). Leo Leppänen's work has been supported by the European Union's Horizon 2020 research and innovation program under grant 825153 (EMBEDDIA).

References

Imen Akermi, Johannes Heinecke, and Frédéric Herledan. 2020. Tansformer based natural language generation for question-answering. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 349–359, Dublin, Ireland. Association for Computational Linguistics.

Ron Artstein and Massimo Poesio. 2005. Bias decreases in proportion to the number of annotators. *Proceedings of FG-MoL*.

Cristina Barros and Elena Lloret. 2015. Input seed features for guiding the generation process: A statistical approach for Spanish. In *Proceedings of the 15th European Workshop on Natural Language Generation*

[22] Relatedly, Freitag et al. (2021) ask annotators to rate the severity of errors in machine translation output, rather than simply marking errors.

(ENLG), pages 9–17, Brighton, UK. Association for Computational Linguistics.

David Beauchemin, Nicolas Garneau, Eve Gaumond, Pierre-Luc Déziel, Richard Khoury, and Luc Lamontagne. 2020. Generating intelligible plumitifs descriptions: Use case application with ethical considerations. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 15–21, Dublin, Ireland. Association for Computational Linguistics.

Anya Belz, Simon Mille, and David M. Howcroft. 2020. Disentangling the properties of human evaluation methods: A classification system to support comparability, meta-evaluation and reproducibility testing. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 183–194, Dublin, Ireland. Association for Computational Linguistics.

Emily M. Bender, Leon Derczynski, and Pierre Isabelle, editors. 2018. *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Santa Fe, New Mexico, USA.

Simon J. Blanchard and Ishani Banerji. 2016. Evidence-based recommendations for designing free-sorting experiments. *Behavior Research Methods*, 48(4):1318–1336.

Dallas Card, Peter Henderson, Urvashi Khandelwal, Robin Jia, Kyle Mahowald, and Dan Jurafsky. 2020. With little power comes great responsibility. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9263–9274, Online. Association for Computational Linguistics.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue - Volume 16*, SIGDIAL '01, page 1–10.

Miruna-Adriana Clinciu, Dimitra Gkatzia, and Saad Mahamood. 2021. It's commonsense, isn't it? demystifying human evaluations in commonsense-enhanced NLG systems. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 1–12, Online. Association for Computational Linguistics.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Ângela Costa, Wang Ling, Tiago Luís, Rui Correia, and Luísa Coheur. 2015. A linguistically motivated taxonomy for machine translation error analysis. *Machine Translation*, 29(2):127–161.

Kees van Deemter and Ehud Reiter. 2018. Lying and computational linguistics. In Jörg Meibauer, editor, *The Oxford Handbook of Lying*. Oxford University Press.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2185–2194, Hong Kong, China. Association for Computational Linguistics.

Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A. Smith, and Yejin Choi. 2021. Scarecrow: A framework for scrutinizing machine text.

Heidi C. Dulay, Marina K. Burt, and Stephen D. Krashen. 1982. *Language two*. New York : Oxford University Press.

Ondřej Dušek, David M. Howcroft, and Verena Rieser. 2019. Semantic noise matters for neural natural language generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 421–426, Tokyo, Japan. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. *Computer Speech & Language*, 59:123–156.

Virginia F Flack, AA Afifi, PA Lachenbruch, and HJA Schouten. 1988. Sample size determinations for the two rater kappa statistic. *Psychometrika*, 53(3):321–325.

Karën Fort, Gilles Adda, and K. Bretonnel Cohen. 2011. Last words: Amazon Mechanical Turk: Gold mine or coal mine? *Computational Linguistics*, 37(2):413–420.

Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021. Experts, errors, and context: A large-scale study of human evaluation for machine translation.

Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna Clinciu, Dipanjan Das, Kaustubh D. Dhole, Wanyu Du, Esin Durmus, Ondřej Dušek, Chris Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, Harsh Jhamtani, Yangfeng Ji, Shailza Jolly, Mihir Kale, Dhruv Kumar, Faisal Ladhak, Aman Madaan,

Mounica Maddela, Khyati Mahajan, Saad Mahamood, Bodhisattwa Prasad Majumder, Pedro Henrique Martins, Angelina McMillan-Major, Simon Mille, Emiel van Miltenburg, Moin Nadeem, Shashi Narayan, Vitaly Nikolaev, Rubungo Andre Niyongabo, Salomey Osei, Ankur Parikh, Laura Perez-Beltrachini, Niranjan Ramesh Rao, Vikas Raunak, Juan Diego Rodriguez, Sashank Santhanam, João Sedoc, Thibault Sellam, Samira Shaikh, Anastasia Shimorina, Marco Antonio Sobrevilla Cabezudo, Hendrik Strobelt, Nishant Subramani, Wei Xu, Diyi Yang, Akhila Yerukola, and Jiawei Zhou. 2021. The gem benchmark: Natural language generation, its evaluation and metrics.

Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2019. Explaining explanations: An overview of interpretability of machine learning. In *Proceedings - 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, DSAA 2018*.

Dimitra Gkatzia and Saad Mahamood. 2015. A Snapshot of NLG Evaluation Practices 2005-2014. In *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*, pages 57–60.

H. P. Grice. 1975. *Logic and Conversation*, pages 41 – 58. Brill, Leiden, The Netherlands.

Tamarinde L. Haven and Dr. Leonie Van Grootel. 2019. Preregistering qualitative research. *Accountability in Research*, 26(3):229–244. PMID: 30741570.

Ryuichiro Higashinaka, Kotaro Funakoshi, Masahiro Araki, Hiroshi Tsukahara, Yuka Kobayashi, and Masahiro Mizukami. 2015a. Towards taxonomy of errors in chat-oriented dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 87–95, Prague, Czech Republic. Association for Computational Linguistics.

Ryuichiro Higashinaka, Masahiro Mizukami, Kotaro Funakoshi, Masahiro Araki, Hiroshi Tsukahara, and Yuka Kobayashi. 2015b. Fatal or not? finding errors that lead to dialogue breakdowns in chat-oriented dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2243–2248, Lisbon, Portugal. Association for Computational Linguistics.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Nancy Ide and James Pustejovsky, editors. 2017. *Handbook of linguistic annotation*. Springer. ISBN 978-94-024-1426-4.

Taichi Kato, Rei Miyata, and Satoshi Sato. 2020. BERT-based simplification of Japanese sentence-ending predicates in descriptive text. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 242–251, Dublin, Ireland. Association for Computational Linguistics.

Klaus Krippendorff. 1970. Bivariate agreement coefficients for reliability of data. *Sociological methodology*, 2:139–150.

Klaus Krippendorff. 2011. Agreement and information in the reliability of coding. *Communication Methods and Measures*, 5(2):93–112.

Klaus Krippendorff. 2018. *Content Analysis: An Introduction to Its Methodology*, fourth edition edition. SAGE Publications, Inc.

Klaus Krippendorff, Yann Mathet, Stéphane Bouvry, and Antoine Widlöcher. 2016. On the reliability of unitizing textual continua: Further developments. *Quality & Quantity*, 50(6):2347–2364.

Zewang Kuanzhuo, Li Lin, and Zhao Weina. 2020. SimpleNLG-TI: Adapting SimpleNLG to Tibetan. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 86–90, Dublin, Ireland. Association for Computational Linguistics.

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, and Emiel Krahmer. 2021. Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech & Language*, 67:101–151.

Saad Mahamood and Ehud Reiter. 2012. Working with clinicians to improve a patient-information NLG system. In *INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference*, pages 100–104.

Alessandro Mazzei. 2015. Translating Italian to LIS in the rail stations. In *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*, pages 76–80, Brighton, UK. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Emiel van Miltenburg and Desmond Elliott. 2017. Room for improvement in automatic image description: an error analysis. *CoRR*, abs/1704.04198.

Emiel van Miltenburg, Chris van der Lee, and Emiel Krahmer. 2021. Preregistering NLP research. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*,

pages 613–623, Online. Association for Computational Linguistics.

Emiel van Miltenburg, Wei-Ting Lu, Emiel Krahmer, Albert Gatt, Guanyi Chen, Lin Li, and Kees van Deemter. 2020. Gradations of error severity in automatic image descriptions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 398–411, Dublin, Ireland. Association for Computational Linguistics.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229.

Robert (Munro) Monarch. 2021. *Human-in-the-Loop Machine Learning*. Manning Publications Co., Shelter Island, New York. ISBN 9781617296741.

Adrian Muscat and Anja Belz. 2015. Generating descriptions of spatial relations between objects in images. In *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*, pages 100–104, Brighton, UK. Association for Computational Linguistics.

Kimberly A. Neuendorf. 2017. *The Content Analysis Guidebook*. SAGE Publications. Second edition, ISBN 9781412979474.

Jason Obeid and Enamul Hoque. 2020. Chart-to-text: Generating natural language descriptions for charts by adapting the transformer model. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 138–147, Dublin, Ireland. Association for Computational Linguistics.

Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. 2021. Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4812–4829, Online. Association for Computational Linguistics.

Maja Popović. 2020. Informative manual evaluation of machine translation output. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5059–5069, Barcelona, Spain (Online). International Committee on Computational Linguistics.

James Pustejovsky and Amber Stubbs. 2012. *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*. " O'Reilly Media, Inc.".

Ehud Reiter. 2018. A structured review of the validity of BLEU. *Computational Linguistics*, 44(3):393–401.

Ehud Reiter, Roma Robertson, A. Scott Lennox, and Liesl Osman. 2001. Using a randomised controlled clinical trial to evaluate an NLG system. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 442–449, Toulouse, France. Association for Computational Linguistics.

Ehud Reiter, Roma Robertson, and Liesl M. Osman. 2003. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144(1):41–58.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Anastasia Shimorina, Yannick Parmentier, and Claire Gardent. 2021. An error analysis framework for shallow surface realisation. *Transactions of the Association for Computational Linguistics*, 9.

M. S. Silberman, B. Tomlinson, R. LaPlante, J. Ross, L. Irani, and A. Zaldivar. 2018. Responsible research with crowds: Pay crowdworkers at least minimum wage. *Commun. ACM*, 61(3):39–41.

Julius Sim and Chris C Wright. 2005. The Kappa Statistic in Reliability Studies: Use, Interpretation, and Sample Size Requirements. *Physical Therapy*, 85(3):257–268.

Symon Stevens-Guille, Aleksandre Maskharashvili, Amy Isard, Xintong Li, and Michael White. 2020. Neural NLG for methodius: From RST meaning representations to texts. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 306–315, Dublin, Ireland. Association for Computational Linguistics.

Mariët Theune, Ruud Koolen, and Emiel Krahmer. 2010. Cross-linguistic attribute selection for REG: Comparing Dutch and English. In *Proceedings of the 6th International Natural Language Generation Conference*.

Craig Thomson and Ehud Reiter. 2020. A gold standard methodology for evaluating accuracy in data-to-text systems. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 158–168, Dublin, Ireland. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Hung-Wen Yeh, Byron J Gajewski, David G Perdue, Angel Cully, Lance Cully, K Allen Greiner, Won S Choi, and Christine Makosy Daley. 2014. Sorting it out: Pile sorting as a mixed methodology for exploring barriers to cancer screening. *Quality & quantity*, 48(5):2569–2587.

A Annotation

A.1 Procedure and definitions

We annotated all papers from INLG2010, ENLG2015, and INLG2020 in two rounds. Round 1 was an informal procedure where we generally checked whether the papers mentioned any errors at all (broadly construed, without defining the term 'error'). Following this, we determined our formal annotation procedure, based on the example papers: first check if the paper is amenable. If so, check if it (a) mentions any errors in the output or (b) contains an error analysis. We used the following definitions:

Amenable A paper is amenable to an error analysis if one of its primary contributions is presenting an NLG system that produces some form of output text. So, NLG experiments are amenable to an error analysis, while survey papers are not.

Error Errors are (countable) instances of something that is wrong about the output.

Error mention An 'error mention' is a reference to such an instance or a class of such instances.

Error analysis Error analyses are defined as formalised procedures through which annotators identify and categorise errors in the output.

A.2 Discussion points

The most discussion took place on the topic of amenability. Are papers that just generate prepositions (Muscat and Belz, 2015) or attributes for referring expressions (Theune et al., 2010) amenable to error analysis? And what about different versions of SimpleNLG? (E.g., Kuanzhuo et al. 2020.) Although these topics feel different from, say, data-to-text systems, we believe it should be possible to carry out an error analysis in these contexts as well. In the end, amenability for us is just an artificial construct to address the (potential) criticism that we cannot just report the amount of error analyses as a proportion of all *NLG papers. As such, our definition for amenability is just a quick heuristic. Determining whether a paper really benefits from an error analysis is a more complex issue, that depends on many contextual factors.

B Papers containing error analyses

Below is a brief summary of the error analyses that we found in our annotation study.

1. Barros and Lloret (2015) investigate the use of different seed features for controlled neural NLG. They analysed all the outputs of their model, and categorised them based on existing lists of common grammatical errors and drafting errors.

2. Akermi et al. (2020) explore the use of pre-trained transformers for question-answering. They conducted a human evaluation study, asking 20 native speakers to indicate the presence of errors in the outputs of a French and English system. These errors were categorised as: *extra words, grammar, missing words, wrong preposition, word order*.

3. Beauchemin et al. (2020) aim to generate explanations of *plumitifs* (dockets), based on the text of the dockets themselves. Following the identification of different errors (defined by the authors as "the lack of realizing a specific part (accused, plaintiff or list of charges paragraphs), instead of evaluating the textual generation," they trace the source of the error back to either an earlier information extraction step, or to the generation procedure.

4. Kato et al. (2020) present a BERT-based approach to simplify Japanese sentence-ending predicates. They took a bottom-up approach to classify the 140 cases where their model could not generate any acceptable cases. The authors then relate the error types to different stages of the generation process, and to the general architecture of their system.

5. Obeid and Hoque (2020) present a neural NLG model for automatically providing natural language descriptions of information visualisations (i.e., charts). They manually assessed 50 output examples, and highlighted the different errors in the text. The authors find that, despite their efforts to prevent it, their model still suffers from hallucination. They identify two kinds of hallucination: either the model associates an existing value with the wrong data point, or it simply predicts an irrelevant token.

A **notable exception** is the paper by Thomson and Reiter (2020), who carry out an error analysis of existing output data from three different systems. This paper was not considered amenable, because

it does not present an NLG system of its own, and thus it was not included in our counts. But even if we were to count this paper among the error analyses, the trend remains the same: very few papers discuss errors in NLG output.

What can Neural Referential Form Selectors Learn?

Guanyi Chen[♠]*, **Fahime Same**[♡]*, and **Kees van Deemter**[♠]
[♠]Department of Information and Computing Sciences, Utrecht University
[♡]Department of Linguistics, University of Cologne
g.chen@uu.nl, f.same@uni-koeln.de, c.j.vandeemter@uu.nl

Abstract

Despite achieving encouraging results, neural Referring Expression Generation models are often thought to lack transparency. We probed neural Referential Form Selection (RFS) models to find out to what extent the linguistic features influencing the RE form are learnt and captured by state-of-the-art RFS models. The results of 8 probing tasks show that all the defined features were learnt to some extent. The probing tasks pertaining to referential status and syntactic position exhibited the highest performance. The lowest performance was achieved by the probing models designed to predict discourse structure properties beyond the sentence level.

1 Introduction

Referring Expression Generation (REG) is one of the main stages of classic Natural Language Generation (NLG) pipeline (Reiter and Dale, 2000; Krahmer and van Deemter, 2012; van Deemter, 2016). REG studies are concerned with two different tasks. The goal of the classic REG task (also called one-shot REG), is to find a set of attributes to single out a referent from a set of competing referents. The second REG task (henceforth discourse REG) is concerned with the generation of referring expressions (RE) in discourse context. Belz and Varges (2007) phrase it as follows: *Given an intended referent and a discourse context, how do we generate appropriate referential expressions (REs) to refer to the referent at different points in the discourse?*

Classic discourse REG was usually understood as a two-step procedure. In the first step, the referential form (RF, i.e, the syntactic type) is determined. For instance, when referring to Joe Biden at a given point in a discourse, the first step is to decide whether to use a proper name ("*Joe Biden*"), a description ("*the president of the USA*"), a demonstrative ("*this person*") or a pronoun ("*he*"). The second step is to determine the RE content, that is, to choose between all the different ways in which a given form can be realised. For instance, to generate a description of Joe Biden, one needs to decide whether to only mention his job (e.g., *The president* entered the Oval Office.), or to mention the country as well (e.g., *The president of the United states* arrived in Cornwall for the G7 Summit.)

In earlier works, computational linguists linked REG to linguistic theories and built discourse REG systems on the basis of linguistic features. For example, Henschel et al. (2000) investigated the impact of 3 linguistic features namely recency, subjecthood, and discourse status on pronominalization, i.e. deciding whether the RE should be realised as a pronoun. Using these features, they used the notion of *local focus* as a criterion for detecting the set of referents that can be pronominalised. The same holds for feature-based models (see Belz et al. (2010) for an overview) where models are trained on linguistically encoded data.

More recently, a number of neural network-based REG models have been presented (Castro Ferreira et al., 2018a; Cao and Cheung, 2019; Cunha et al., 2020), where they propose to generate REs in an End2End manner without any feature engineering. They all used a benchmark dataset called WebNLG. These models generally follow the sequence-to-sequence framework (Sutskever et al., 2014), where there is an encoder for encoding the given discourse, and a decoder responsible for generating REs using the encoded information. The evaluation results suggested that these neural methods perform well not only for selecting the proper RFs, but also for producing fluent REs. However, it was unclear to what extent these neural models can encode linguistic features.

To conduct model inspection, we introduce a

* Equal contribution

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 154–166,
Aberdeen, Scotland, UK, 20-24 September 2021.

Triples: (AWH_Engineering_College, country, India), (Kerala, leaderName, Kochi), (AWH_Engineering_College, academicStaffSize, 250), (AWH_Engineering_College, state, Kerala), (AWH_Engineering_College, city, "Kuttikkattoor"), (India, river, Ganges)
Text: AWH Engineering College is in Kuttikkattoor, India in the state of Kerala. The school has 250 employees and Kerala is ruled by Kochi. The Ganges River is also found in India.
Delexicialised Text: **Pre-context**: AWH_Engineering_College is in "Kuttikkattoor" , India in the state of Kerala . **Target Entity**: AWH_Engineering_College **Pos-context**: has 250 employees and Kerala is ruled by Kochi . The Ganges River is also found in India .

Table 1: An example data from the WebNLG corpus. In the delexicalised text, every entity is underlined.

series of probing tasks. Using probing tasks is a well-established method to analyse whether a model's latent representation encodes specific information. This approach has been widely used for analysing models in machine translation (Belinkov et al., 2017), language modelling (Giulianelli et al., 2018), relation extraction (Alt et al., 2020), and so on. Additionally, there had been various works on coreference resolution and bridging anaphora (Sorodoc et al., 2020; Pandit and Hou, 2021) which, similar to this paper, target the understanding of reference. More precisely, for a probing task, a diagnostic classifier is trained on representations from the model. Its performance embodies how well those representations encode the information associated with the probing task. The aim of this paper is to understand what linguistic features neural models encode when modelling REs.

Our main focus is on the encoding of linguistic features in the representations. In the linguistic tradition, the majority of RE production studies focus on Referential Form Selection (RFS), rather than RE content realisation. Our focus in the present work is likewise on RFS. To tackle RFS, we adopt the state-the-of-the-art neural REG model of (Castro Ferreira et al., 2018a). Additionally, to make comparison, we also propose (1) a strong baseline that uses only a single encoder (while Castro Ferreira et al. (2018a) used multiple encoders); and (2) to leverage pre-trained word embeddings (e.g., GloVe) or language models (e.g., BERT).

Therefore, in this paper, we first introduce the task of RFS on the basis of WebNLG corpus, and propose a number of neural models to tackle the task. Subsequently, we introduce 8 probing tasks, each of which is associated with a linguistic feature influencing the choice of RF. We examine our RFS models on these probing tasks in order to interpret and explain their behaviour. The code of each RFS model and the probing classifier is available at: github.com/a-quei/probe-neuralreg.

2 Background

2.1 Discourse REG

Given a text whose REs have not yet been generated, and given the intended referent for each of these REs, the discourse REG task is to build an algorithm that generates all these REs. So far, this task has attracted many research efforts (e.g., Hendrickx et al. (2008); Greenbacker and McCoy (2009)) and it has been used in the GREC shared tasks (Belz et al., 2010).

More recently, this task was formulated into a format that goes together well with deep learning: Castro Ferreira et al. (2018a) introduced the End2End REG task, built a corresponding dataset based on WebNLG (Castro Ferreira et al., 2018b), and constructed NeuralREG models.

The WebNLG corpus was originally designed to assess the performance of NLG systems (Gardent et al., 2017). Each sample in this corpus contains a knowledge base described by a Resource Description Framework (RDF) triple (Table 1). Castro Ferreira et al. (2018a) and Castro Ferreira et al. (2018b) enriched and delexicalised the corpus to fit the discourse REG task. Table 1 shows a text created from a RDF, and its corresponding delexicalised version.

Taking the delexicalised text in Table 1 as an example, given the entity "*AWH_Engineering_College*", REG chooses a RE based on that entity and its pre-context ("*AWH_Engineering_College is in "Kuttikkattoor" , India in the state of Kerala .* ") and its pos-context ("*has 250 employees and Kerala is ruled by Kochi . The Ganges River is also found in India .*").

2.2 Factors that Influence RE Production

Languages display a large inventory of expressions for referring to entities (von Heusinger and Schumacher, 2019). In linguistics, the realisation choice

a speaker makes has been associated with the accessibility, i.e. activation of mental representations of a referent at a particular point in discourse: attenuated forms such as pronouns are often used to refer to highly accessible or highly activated referents, while richer forms such as descriptions and proper names are employed in referring to less accessible ones (Ariel, 1990; Gundel et al., 1993). Due to the central role of referring in communication, a wealth of research has tried to assess the influence of different features modulating the accessibility of a referent. von Heusinger and Schumacher (2019) refer to these features as *prominence-lending cues*, meaning that they increase the prominence status of their respective referents to some extent. In this section, we merely talk about the ones which will be taken up in our probing experiments, and will not further discuss cues such as animacy (Fukumura and van Gompel, 2011), competition (Arnold, 2010) and coherence relations (Kehler et al., 2008).

Referential status or *givenness* has been widely discussed in the literature (see Chafe (1976); Prince (1981)). When a new character is introduced into the discourse, the chance that this happens by means of a pronoun is slim (unless the referent is situationally given). Pronouns are reserved for referring to previously introduced (or given) referents.

Recency, another well-studied cue, is defined as the distance between the target referent and its antecedent. If a referent is not too far apart from its antecedent, then reduced forms are typically employed to refer to it.

There are also intra-clausal cues such as *grammatical role* (Brennan, 1995) and *thematic role* (Arnold, 2001) which impact the prominence status of referents. For instance, the subject of a sentence is perceived to be more prominent than the object.

Discourse-structural features affect the organisational aspects of discourse. Centering-based theories (Grosz et al., 1995) often use the notion of local focus to account for pronominalisation. *Local focus* takes the current and previous utterance into account. *Global focus*, on the other hand, situates a referent in a larger space, namely the whole text or a discourse segment (Hinterwimmer, 2019). Concepts such as the importance of a referent or familiarity are associated with the global prominence status of entities (Siddharthan et al., 2011).

Type	Classes
4-Way	Demonstrative, Description, Proper Name, Pronoun
3-Way	Description, Proper Name, Pronoun
2-Way	Non-pronominal, Pronominal

Table 2: 3 different types of RF classification.

3 Neural Referential Form Selection

In this section, we define the task of RFS built on the WebNLG dataset, and introduce a number of NeuralRFS models.

3.1 The RFS Task

Akin to REG, given the previous context $x^{(pre)} = \{w_1, w_2, ..., w_{i-1}\}$ (where w is either a word or a delexicalised entity label), the target referent $w^{(r)} = \{w_i\}$, and the post context $w^{(pos)} = \{w_i, w_{i+1}, ..., w_n\}$, a RFS algorithm aims at finding the proper RF $\hat{f}$ from a set of K candidate RFs $\mathcal{F} = \{f_k\}_{k=1}^{K}$.

Regarding the possible RFs for the RFS task, we test 3 different classifications, depicted in Table 2. Due to the small number of demonstrative noun phrases in the dataset, we decided to also conduct a 3-way classification in which descriptions and demonstratives are merged. Also, most emphasis in the linguistic literature is on the pronominalisation issue. Therefore, we also included a 2-way classification task in the study.

As stated, the main goal of the paper is to understand which linguistic features are encoded by RFS neural models. Additionally, we were curious whether models trained solely for pronominalisation capture different contextual features in comparison with the other two classifications.

3.2 NeuralRFS Models

We build NeuralRFS models by (1) adopting the best NeuralREG model from Castro Ferreira et al. (2018a), and (2) proposing a new alternative which is simpler, and can easier incorporate pre-trained representations.

ConATT. We adopt the CATT model from Castro Ferreira et al. (2018a), which achieves the best performance on REG among the models they tested in their study. Given the inputs, we first use Bidirectional GRU (BiGRU, Cho et al., 2014) to encode $x^{(pre)}$ as well as $x^{(pos)}$. Formally, for each $k \in [pre, pos]$, we encode $x^{(k)}$ to $h^{(k)}$ with a BiGRU: $h^{(k)} = \text{BiGRU}(x^{(k)})$. Subsequently, differ-

Model	4-way			3-way			2-way		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
XGBoost	53.77	51.98	51.55	71.27	69.24	68.34	86.64	82.76	84.57
c-RNN	68.79	62.95	64.96	84.49	82.52	**83.63**	90.31	88.01	89.09
+GloVe	**69.10**	**63.90**	**65.40**	84.29	**82.55**	83.30	89.33	88.02	88.63
+BERT	62.63	61.80	62.15	83.02	81.44	82.15	**90.98**	88.00	**89.42**
ConATT	67.42	62.39	64.07	**85.04**	82.21	83.53	89.30	**89.19**	89.23
+GloVe	65.98	62.49	63.67	83.62	81.41	82.45	89.60	88.06	88.80

Table 3: Evaluation results of our RFS systems on WEBNLG. Best results are **boldfaced**, whereas the second best results are underlined.

ent from Castro Ferreira et al. (2018a), we encode $h^{(k)}$ into the context representation $c^{(k)}$ using self-attention (Yang et al., 2016). Concretely, given the total N steps in $h^{(k)}$, we first calculate the attention weight $\alpha_j^{(k)}$ at each step j by:

$$e_j^{(k)} = v_a^{(k)T} \tanh(W_a^{(k)} h_j^{(k)}), \quad (1)$$

$$\alpha_j^{(k)} = \frac{\exp(e_j^{(k)})}{\sum_{n=1}^{N} \exp(e_n^{(k)})}, \quad (2)$$

where v_a is the attention vector and W_a is the weight in the attention layer. The context representation of $x^{(k)}$ is then the weighted sum of $h^{(k)}$:

$$c^{(k)} = \sum_{j=1}^{N} \alpha_j^{(k)} h^{(k)}. \quad (3)$$

After obtaining $c^{(pre)}$ and $c^{(pos)}$, we concatenate them with the target entity embedding $x^{(r)}$, and pass it through a feed forward network to obtain the final representation:

$$R = \text{ReLU}(W_f[c^{(pre)}, x^{(r)}, c^{(pos)}]), \quad (4)$$

where W_f is the weights in the feedforward layer. R is also used as the input of the probing classifiers (section 4). R is then fed for making the final prediction:

$$P(f|x^{(pre)}, x^{(r)}, x^{(pos)}) = \text{Softmax}(W_c R), \quad (5)$$

where W_c is the weight in the output layer.

c-RNN. In addition to ConATT, we also try a simpler yet effective structure, which uses only a single BiGRU. We name the framework it follows as the centred recurrent neural networks (henceforth c-RNN). Specifically, instead of using two separate BiGRUs to encode pre- and pos-contexts, we first concatenate $x^{(pre)}$, $x^{(r)}$, and $x^{(pos)}$, and then encode them together:

$$h = \text{BiGRU}([x^{(pre)}, x^{(r)}, x^{(pos)}]). \quad (6)$$

Suppose that the target entity is in position i of the concatenated sequence, we extract the i-th representation from h_i for obtaining $R = \text{ReLU}(W_f h_i)$. After obtaining R, the rest of the procedure is the same as ConATT.

Pre-training. As a secondary objective of this study, we want to see whether RFS can benefit from pre-trained word embeddings and language models, whose effectiveness has not yet been explored in REG[1]. For both c-RNN and ConATT, we try the GloVe embeddings (Pennington et al., 2014) to see how pre-trained word embeddings contribute to the choice of RF. For c-RNN, we try to stake it on the BERT (Devlin et al., 2019) model. In order to let BERT better encode the delexicalised entity labels, we first re-train BERT as a masked language model on the training data of WebNLG. We then freeze the parameters of BERT and use the model to encode the input, which is then fed into c-RNN[2].

Machine Learning (ML) based Model We used XGBoost (Chen and Guestrin, 2016) from the family of Gradient Boosting Decision Trees to train RFS classifiers. 5-fold-cross-validation was used to train the models. The classifiers were first trained on a wide range of features obtained from the WebNLG corpus (16 features). After running a variable importance analysis, we selected a subset of features for the final models. The detailed list of features are presented in Appendix A.

[1]Previously, only Cao and Cheung (2019) used pre-trained embeddings, but no ablation study was done.

[2]We also explored other ways of using BERT, such as using only BERT plus a feed forward layer to obtain h, or not freezing parameters of BERT while training. The resulting models had low performance in all cases.

Figure 1: Confusion Matrices for 4-way classification results of XGBoost (left) and c-RNN+GloVe (right), where PRO, PN, DES, and DEM are pronoun, proper name, description and demonstrative respectively.

3.3 Evaluation

Implementation Details. We tuned hyper-parameters of each of our models on the development set and chose the setting with the best macro F1 score. For the BERT model, we used the cased BERT-BASE[3] and added all entity labels into the vocabulary to avoid tokenisation. When re-training BERT on WebNLG, we set the masking probability to 0.15 and trained it for 25 epochs.

For the XGBoost models, we set the learning rate to 0.05, the minimum split loss to 0.01, the maximum depth of a tree to 5, and the sub-sample ratio of the training instances to 0.5.

We report the macro averaged precision, recall, and F1 on the test set. We run each model for 5 times, and report the averaged performance. As for the dataset, we use the v1.5 of WebNLG (Castro Ferreira et al., 2019) and use only seen entities.

Results. Table 3 shows the results of different classification tasks. Generally, all neural variants outperform the machine learning baseline. The performance difference is small in the case of binary classification, while it is much bigger for 3- and 4-way classifications. This is because the 2-way classification (i.e., pronominalisation) has clearly less complexity than the other two alternatives, and, thus, the feature set used by the baseline results in almost similar outcomes to neural models.

Comparing neural variants to each other, the results show that the simpler c-RNN wins over ConATT in 4-way classification, and has on par performance with ConATT for 3- and 2-way classifications. One possible explanation is that ConATT first breaks down the input into three pieces (i.e., the target entity as well as pre- and pos-context), encodes them separately, and merges the encoded representations back before being sent to make predictions. This "divide and merge" procedure might hinder the model from learning some useful information.

Regarding the effectiveness of incorporating pre-trained models, GloVe embeddings have positive impact on c-RNN only in case of 4-way predictions, and have no contribution to 2- and 3-way classifications. Moreover, it has negative effect on ConATT: the performance diminishes when GloVe is used. It is surprising to see that in case of c-RNN, BERT has negative effect on 4- and 3-way predictions (the F1 score reduced from 64.86 and 83.63 to 62.15 and 82.15 respectively). For pronominalisation, BERT slightly boosts the performance (from 89.09 to 89.42), but this boost is not as much as BERT's boosting effect on other NLP tasks. This is probably because although BERT was re-trained on WebNLG delexicalised sentences, the entity labels still function as noise for BERT.

To obtain insights into the behaviours of the deep learning and classic ML-based models for RFS, we depict the confusion matrices of XGBoost and the best performing neural model c-RNN+GloVe in Figure 1 for the 4-way classification. The confusion matrices suggest that both models do a good job in selecting pronouns and proper names (that is why the performance difference in the 2-way classification is small), and both perform poorly in choosing demonstratives (probably due to the fact that demonstratives are extremely infrequent in WebNLG). The main difference between the two models is in distinguishing proper names from descriptions. The XGBoost model wrongly predicted the descriptions as proper names in 62.58% of the cases, while the neural c-RNN+GloVe

[3] huggingface.co/bert-base-cased

model did this wrong prediction in 20.18% of the times. This difference in the performance of the two models might be because the neural models learnt some useful features from the discourse which are not covered in our feature engineering procedure. Furthermore, after looking into the WebNLG dataset, we noticed that various RE cases are annotated incorrectly. For example, WebNLG annotates "*United States*" as a proper name, and "*the United States*" as a description. The incorrect annotations might increase the confusion between choosing description and proper name in both `XGBoost` and `c-RNN+GloVe`.

4 Probing RFS models

We use a logistic regression classifier as our probing classifier. Concretely, for each input, we first use a model discussed in section 3 to obtain its representation R. As mentioned in section 3, we ran each model five times and reported their averaged scores. For the probing tasks, we use the representations of the models with the best RFS performance on the development set.

4.1 Probing Tasks

Following our observations in section 2.2, we formulate the following probing tasks.

Referential Status. The referential status of the target entity influences the choice of RF in both linguistic (Chafe, 1976; Gundel et al., 1993) and computational studies (Castro Ferreira et al., 2016). In this study, we define referential status on two levels: discourse-level and sentence-level. The former (**DisStat**) has two possible values: (a) discourse-old (i.e., the entity has appeared in the previous discourse) and (b) discourse-new (i.e., the entity has not appeared in the previous discourse). Sentence-level referential status (**SenStat**) also consists of two values: (a) sentence-new (i.e., the RE is the first mention of the entity in the sentence), and (b) sentence-old (i.e., the RE is not the first mention).

Syntactic Position. Entities in subject position are more likely to be pronominalised than in object position (Brennan, 1995; Arnold, 2010). Therefore, in the syntax probing task (henceforth **Syn**), we do binary classification: subject or object.

Recency. Recency has been used as a vital feature in many of the previous REG or RFS systems (Greenbacker and McCoy, 2009; Kibrik et al., 2016). It measures the distance between the target entity and its closest antecedent. There are various ways of estimating the recency of a target entity given its context. We hereby use two measures: (1) the number of sentences between the target entity and its antecedent (**DistAnt**), which consists of four possible values: the entity and its antecedent are (a) in the same sentence, (b) one sentence away, (c) more than one sentence away, and (d) the entity is a first mention (to distinguish first mentions from subsequent mentions). (2) whether there is an intervening referent between the target and its nearest antecedent (**IntRef**) (Greenbacker and McCoy, 2009). In other words, it checks whether the target and the preceding RE are coreferential. This feature has three possible values: (a) the target entity is a first mention, (b) the previous RE refers to the same entity, and (c) the previous RE refers to a different entity. Note that the existence of intervening markables might signal the existence of a competition (if the intervening referent has the same animacy and gender values as the target RE).

Discourse Structure Prominence. As mentioned in section 2, the "organizational" properties of discourse may influence the prominence status of the entities. We introduce three probing tasks capturing different properties of the discourse. (1) *Local prominence* (**LocPro**): The idea of local prominence is coming from Centering Theory (Grosz et al., 1995). It is a hybrid feature of DisStat and Syn. Concretely, we use the implementation of Henschel et al. (2000): an entity is *locally prominent* if it is "discourse-old" and "realised as subject". It is a binary feature with two possible values: (a) locally prominent, and (b) not locally prominent. (2) *Global prominence* (**GloPro**): This feature is based on the notion of global salience in Siddharthan et al. (2011), asking whether the entity is a minor or major referent in the text. According to them, "the frequency features are likely to give a good indication of the global salience of a referent in the document" (p. 820). We define a binary feature in which the most frequent entity in a text is marked as globally prominent. (3) *Meta-prominence* (**MetaPro**): In line with global prominence, we also want to explore to what extent prominence beyond a single text (e.g. on a text collection level) may impact the way people refer. In the context of the current circumstances, the sentence "I received *my vaccine* today" is unambiguous, and the RE *my vaccine* needs no extra modification (e.g. my COVID-19 vaccine); how-

Model	Type	DisStat	SenStat	Syn	DistAnt	IntRef	LocPro	GloPro	MetaPro
Random	-	49.57 (41.83)	33.11 (22.87)	49.65 (48.99)	25.19 (14.90)	33.30 (22.92)	50.05 (49.84)	49.75 (48.02)	25.24 (25.20)
Majority	-	86.91 (46.50)	86.91 (31.00)	61.27 (37.99)	86.91 (23.25)	86.91 (31.00)	56.28 (36.01)	68.49 (40.65)	28.12 (10.97)
c-RNN	4-way	85.16 (84.06)	93.28 (73.72)	94.16 (85.34)	92.84 (53.84)	91.71 (55.43)	83.37 (82.92)	70.62 (56.00)	44.76 (42.32)
	3-way	84.78 (83.72)	92.59 (72.60)	93.50 (83.60)	92.58 (54.78)	91.24 (53.21)	82.17 (81.67)	70.87 (56.70)	45.42 (41.79)
	2-way	88.84 (88.04)	92.77 (73.84)	93.49 (84.00)	92.53 (54.93)	91.01 (52.31)	86.08 (85.69)	71.24 (59.98)	44.32 (41.65)
c-RNN +GloVe	4-way	85.84 (84.85)	93.58 (74.59)	94.56 (87.04)	93.30 (55.67)	92.06 (55.93)	83.71 (83.20)	70.55 (53.53)	44.23 (41.71)
	3-way	85.09 (83.89)	91.89 (67.24)	93.23 (82.48)	91.72 (50.94)	90.92 (51.17)	82.08 (81.44)	70.20 (52.49)	45.58 (42.34)
	2-way	88.88 (88.02)	92.38 (71.25)	93.32 (82.67)	92.25 (53.67)	90.94 (51.43)	85.81 (85.22)	71.78 (63.17)	44.92 (41.03)
c-RNN +BERT	4-way	95.85 (90.64)	94.41 (78.04)	84.05 (82.71)	93.60 (56.91)	92.27 (54.30)	82.03 (81.67)	71.04 (54.24)	45.27 (43.07)
	3-way	94.00 (84.80)	92.74 (72.29)	85.12 (84.08)	92.57 (54.21)	91.28 (53.25)	82.92 (82.53)	71.69 (57.31)	43.64 (42.80)
	2-way	94.59 (87.28)	92.94 (69.69)	85.75 (84.74)	92.50 (54.19)	92.06 (54.88)	83.27 (82.77)	73.80 (63.07)	41.05 (40.75)
ConATT	4-way	94.86 (87.81)	94.12 (77.11)	88.64 (88.00)	93.69 (57.09)	92.11 (55.88)	86.93 (86.34)	72.22 (60.15)	48.37 (46.14)
	3-way	93.91 (84.39)	93.15 (74.19)	87.43 (86.66)	92.93 (55.26)	91.35 (54.09)	85.32 (84.56)	72.61 (60.61)	49.35 (47.47)
	2-way	93.74 (84.20)	92.78 (73.18)	89.01 (88.44)	92.50 (53.98)	91.19 (53.64)	87.05 (86.75)	70.65 (56.39)	44.24 (41.81)
ConATT +GloVe	4-way	94.86 (87.82)	94.10 (77.70)	87.98 (87.24)	93.66 (57.52)	92.10 (55.22)	86.06 (85.69)	71.94 (58.54)	53.19 (49.94)
	3-way	93.79 (84.35)	92.78 (72.83)	89.54 (88.91)	92.59 (54.23)	91.39 (51.96)	87.09 (86.80)	71.91 (59.05)	49.27 (46.36)
	2-way	93.81 (84.38)	92.86 (73.21)	87.69 (86.96)	92.84 (56.14)	91.50 (53.33)	85.61 (85.27)	72.48 (62.46)	44.47 (39.63)

Table 4: Results of each probing task. Results are reported in the format of A(B), where A is the accuracy and B is the macro F1.

ever, a couple of years from now, a richer RE may be needed to refer to the vaccine. The idea behind this exploratory feature is that people might use less semantic content to refer to the referents which are well known outside of the text. Based on the number of mentions of a target entity in the whole WebNLG, four possible values, each of which representing an interval, are assigned to each RE: (a) $[0, 50)$, (b) $[50, 150)$, (c) $[150, 290)$, and (d) $[290, \infty)$. For example, the category $[0, 50)$ contains those entities that occur fewer than 50 times in the corpus.

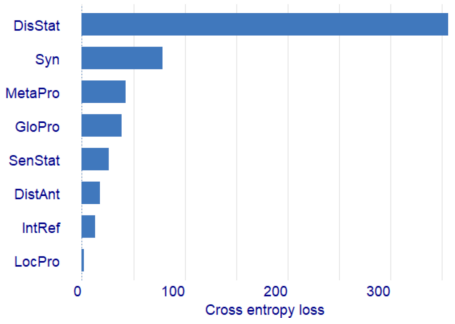


Figure 2: Feature importance of XGBoost classifiers for 4-way predictions. Higher loss shows greater importance of a feature. Results for 2-way and 3-way classification can be found in the Appendix B.

4.2 Importance Analysis

We conducted a feature importance analysis to find out which features used in the probing tasks had the highest contributions to the feature-based ML models. This analysis functions as a sanity check to find out whether the representations have learnt the features contributing the most to the RFS task.

To assess the importance of the features used in the probing tasks, we train XGBoost models, only using features from section 4.1, and calculate

the model-agnostic permutation-based variable importance of each model (Biecek and Burzykowski, 2021). Concretely, we measure the extent to which the performance changes if we remove one of the features. Figure 2 depicts the performance change for each feature. According to the figure, DisStat and Syn contribute the most. LocPro is the least important feature because it is a hybrid combination of DisStat and Syn. Removing it while keeping DisStat and Syn will not hurt the performance of the model a lot. Considering that DisStat and Syn are both highly vital features, LocPro is much more important than what the experiment suggests. In addition to DisStat and Syn probing tasks, we also expect high performance for the LocPro task.

4.3 Probing Results

We mentioned earlier that we conduct probing tasks to find out whether the RFS models' latent representations encode the features mentioned in section 4.1. High performance in probing tasks would indicate that the features are encoded in the latent representations of the models.

We evaluate probing tasks using the accuracy and macro-averaged F1 scores. Each probing classifier was trained 5 times. We report the averaged value. Additionally, we use 2 baselines: (1) `random`: it randomly assigns a label to each input; and (2) `majority`: it assigns the most frequent label in the given probing task to the inputs.

Results of Each Probing Task. Compared to the `random` baseline, all neural models have achieved higher performance on all tasks. (1) Referential status and syntactic position: all models exhibit consistently high performance on DisStat, SenStat, and Syn. This shows that, at least for the WebNLG corpus, all neural models can learn information of referential status and syntactic position; (2) Recency (i.e., DistAnt and IntRef): all models perform worse compared to the referential status and syntax probes. Although they do not have bad accuracy scores, their F1 scores are lower than that of DisStat, SenStat, and Syn, and are closer to the baselines. This finding is consistent with the results of section 4.2, where DistAnt and IntRef were found to be less important (comparing to DisStat and Syn). One possible explanation is that, in the WebNLG corpus, 67% of the documents contain only one sentence, making recency-related features play less role. As another possible explanation, in line with the previous probing works on coreference and bridging anaphora (Sorodoc et al., 2020; Pandit and Hou, 2021), models have more difficulty capturing long-distance properties; (3) Discourse structure prominence: since LocPro is a hybrid of DisStat and Syn, all models handled it to a large degree. Meanwhile, neural models appear to handle GloPro and MetaPro worse than other features since the performance of their corresponding probing tasks is closer to the baselines[4]. These results are in contrast with the importance analysis results, which suggested that both GloPro and MetaPro are important features (ranking 3 and 4 in Figure 2). Learning GloPro and MetaPro requires a model to have an overall understanding of the whole input document or the whole corpus, which the neural models might not be able to acquire.

Comparing `c-RNN` and `ConATT`. In section 3, we concluded that the `c-RNN` model works better than `ConATT` on 4-way RF classification. Nevertheless, when probing, we observed that `ConATT` does a better job in many tasks, including DisStat, LocPro, GloPro, and MetaPro. To understand why, we looked into the WebNLG dataset and found that 86.91% of the REs in WebNLG are first mentions, and 21% of the documents talk about the entity "*United_States*". This suggests that REs in WebNLG are not representative of the realistic use of REs. Therefore, although `ConATT` learns more contextual features, it still has a lower performance. `ConATT`'s better learning of referential status (i.e., DisStat) is probably a benefit of using self-attention, which helps the model capture longer dependencies than RNNs.

The Effect of Pre-training. As mentioned earlier, the secondary objective of this study is to find out whether RFS can benefit from pre-trained word embeddings and language models. The effect of incorporating the `GloVe` embeddings is not significant to `c-RNN` and `ConATT`. The major contribution of `BERT` is helping with learning DisStat (which is, again, probably a result of using self-attention). Akin to the above discussion, since the majority of the entities in WebNLG are first mentions, the increased accuracy boost in the DisStat task is not enough to boost the overall performance of RFS.

Comparing Different RF Classifications. It also appears that models learn different infor-

[4]Note that, for MetaPro, the `Majority` has low F1 score because the distribution of the values of MatePro is balanced.

mation using different label sets (classes). For example, 2-way classification (i.e., pronominalisation) helps `c-RNN` learn more about referential status. But in case of models with attention mechanism (i.e., `ConATT`, `ConATT+GloVe` and `c-RNN+BERT` models), referential status is learnt better in 4-way classification models. Also, in case of `ConATT(+GloVe)`, we observe that more fine-grained classifications help the model learn more about meta prominence (i.e., MetaPro).

5 Conclusion

Our aim is to understand whether neural models capture the features associated with the task of RFS. To this end, we defined 8 probing tasks in which we focused on referential status, syntactic position, recency, and discourse structure. The probing results suggest that the probe classifiers always performed better than the `random` and the `majority` baselines. The performance was consistently good in the tasks associated with referential status, syntax and local prominence.

It is worth noting that probing has its own shortcomings. For instance, on the one hand, low probing performance does not always mean the feature is not encoded, but could also mean that such a feature does not matter to RFS. To mitigate this issue, we conducted a complementary ML-based variable importance analysis; in this analysis, discourse status and syntactic position came out as the factors with the highest contributions. These features were also predicted very well in the probing tasks. However, these results should still be taken with a pinch of salt: the variable importance has been conducted on the ML model and not on the neural models. We cannot be certain that the same features contribute to all the models similarly: a feature might be quite important in the machine learning model, but not as important in the neural models. On the other hand, some researches have questioned the validity of probing methods. They found out that it is difficult to distinguish between "learning the probing task" and "extracting the encoded linguistic information" (Hewitt and Liang, 2019; Kunz and Kuhlmann, 2020) for a probing classifier. This suggests that higher performance of a probing classifier does not necessarily mean more linguistic information has been encoded. This prevents us from directly quantifying *how well* the linguistic information has been learnt using the performance of probing classifiers and requires us to make conclusions more carefully.

From our probing efforts, we conclude that: (1) All neural models have learnt some information about the features associated with the probing tasks, but how well they have learnt this information is yet to be assessed; (2) The WebNLG corpus, which has often been used for the study of discourse REG, is not ideally suitable for studying discourse-related aspects of RFS, because the texts are too short and the majority of REs are first mentions. This leads to bias in the evaluation of RFS and REG algorithms; (3) When it comes to the question of how well a RFS feature can be learnt, it matters what neural architecture and label set are used, and whether the model is pre-trained or not. Using an attention mechanism and more fine-grained label sets help a model learn more information; (4) All models perform poorly in terms of learning those features, such as GloPro and MetaPro, that do not derive from the text itself but from the wider context in which it is written and read. We believe that future models should take these lessons into consideration.

In future, we plan to extend the current study from three angles. First, we plan to conduct experiments on different corpora. The WebNLG corpus used in this study consists predominantly of extremely short documents with an average length of only 1.4 sentences/document; consequently the majority of REs are first mentions. We hope to find a more representative distribution of uses of referring expressions in other corpora such as OntoNotes (Hovy et al., 2006), which contain longer texts. Secondly, we plan to conduct experiments on other languages than English, in particular ones that favour zero pronouns (e.g., Chinese (Chen et al., 2018)), because these pose new challenges for the task of RFS. Thirdly, we plan to design new probing tasks on the basis of other factors that could influence RFS, such as animacy, competition and positional attributes (see Same and van Deemter (2020) for an overview).

Acknowledgements

We thank the anonymous reviewers for their helpful comments. Guanyi Chen is supported by China Scholarship Council (No.201907720022). Fahime Same is supported by the German Research Foundation (DFG)– Project-ID 281511265 – SFB 1252 "Prominence in Language".

References

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. 2020. Probing linguistic features of sentence-level representations in neural relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1534–1545, Online. Association for Computational Linguistics.

Mira Ariel. 1990. *Accessing Noun-Phrase Antecedents*. Routledge.

Jennifer E Arnold. 2001. The effect of thematic roles on pronoun use and frequency of reference continuation. *Discourse processes*, 31(2):137–162.

Jennifer E Arnold. 2010. How speakers refer: The role of accessibility. *Language and Linguistics Compass*, 4(4):187–203.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Anja Belz, Eric Kow, Jette Viethen, and Albert Gatt. 2010. Generating referring expressions in context: The GREC task evaluation challenges. In *Empirical Methods in Natural Language Generation: Data-oriented Methods and Empirical Evaluation*, volume 5790 of *Lecture Notes in Computer Science*, pages 294–327. Springer.

Anja Belz and Sebastian Varges. 2007. Generation of repeated references to discourse entities. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*, pages 9–16, Saarbrücken, Germany. DFKI GmbH.

Przemyslaw Biecek and Tomasz Burzykowski. 2021. *Explanatory model analysis: explore, explain, and examine predictive models*. CRC Press.

Susan E Brennan. 1995. Centering attention in discourse. *Language and Cognitive processes*, 10(2):137–167.

Meng Cao and Jackie Chi Kit Cheung. 2019. Referring expression generation using entity profiles. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3163–3172, Hong Kong, China. Association for Computational Linguistics.

Thiago Castro Ferreira, Emiel Krahmer, and Sander Wubben. 2016. Towards more variation in text generation: Developing and evaluating variation models for choice of referential form. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 568–577, Berlin, Germany. Association for Computational Linguistics.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 552–562, Hong Kong, China. Association for Computational Linguistics.

Thiago Castro Ferreira, Diego Moussallem, Ákos Kádár, Sander Wubben, and Emiel Krahmer. 2018a. NeuralREG: An end-to-end approach to referring expression generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1969, Melbourne, Australia. Association for Computational Linguistics.

Thiago Castro Ferreira, Diego Moussallem, Emiel Krahmer, and Sander Wubben. 2018b. Enriching the WebNLG corpus. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 171–176, Tilburg University, The Netherlands. Association for Computational Linguistics.

Wallace Chafe. 1976. Givenness, contrastiveness, definiteness, subjects, topics, and point of view. *Subject and topic*.

Guanyi Chen, Kees van Deemter, and Chenghua Lin. 2018. Modelling pro-drop with the rational speech acts model. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 159–164, Tilburg University, The Netherlands. Association for Computational Linguistics.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA. Association for Computing Machinery.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Rossana Cunha, Thiago Castro Ferreira, Adriana Pagano, and Fabio Alves. 2020. Referring to what you know and do not know: Making referring expression generation models generalize to unseen entities. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2261–

2272, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Kees van Deemter. 2016. *Computational models of referring: a study in cognitive science*. MIT Press.

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

Kumiko Fukumura and Roger PG van Gompel. 2011. The effect of animacy on the choice of referring expression. *Language and cognitive processes*, 26(10):1472–1504.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248, Brussels, Belgium. Association for Computational Linguistics.

Charles Greenbacker and Kathleen McCoy. 2009. UDel: generating referring expressions guided by psycholinguistic findings. In *Proceedings of the 2009 Workshop on Language Generation and Summarisation*, pages 101–102. Association for Computational Linguistics.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Jeanette K Gundel, Nancy Hedberg, and Ron Zacharski. 1993. Cognitive status and the form of referring expressions in discourse. *Language*, pages 274–307.

Iris Hendrickx, Walter Daelemans, Kim Luyckx, Roser Morante, and Vincent Van Asch. 2008. CNTS: Memory-based learning of generating repeated references. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 194–95, Salt Fork, Ohio, USA. Association for Computational Linguistics.

Renate Henschel, Hua Cheng, and Massimo Poesio. 2000. Pronominalization revisited. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 306–312. Association for Computational Linguistics.

Klaus von Heusinger and Petra B Schumacher. 2019. Discourse prominence: Definition and application. *Journal of Pragmatics*, 154:117–127.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Stefan Hinterwimmer. 2019. Prominent protagonists. *Journal of Pragmatics*, 154:79–91.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA. Association for Computational Linguistics.

Andrew Kehler, Laura Kertz, Hannah Rohde, and Jeffrey L Elman. 2008. Coherence and coreference revisited. *Journal of semantics*, 25(1):1–44.

Andrej A. Kibrik, Mariya V. Khudyakova, Grigory B. Dobrov, Anastasia Linnik, and Dmitrij A. Zalmanov. 2016. Referential choice: Predictability and its limits. *Frontiers in Psychology*, 7:1429.

Emiel Krahmer and Kees van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Jenny Kunz and Marco Kuhlmann. 2020. Classifier probes may just learn from linear context features. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5136–5146, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Onkar Pandit and Yufang Hou. 2021. Probing for bridging inference in transformer language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4153–4163, Online. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Ellen F Prince. 1981. Towards a taxonomy of given-new information. *Radical pragmatics*.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press.

Fahime Same and Kees van Deemter. 2020. A linguistic perspective on reference: Choosing a feature set for generating referring expressions in context. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4575–4586, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2011. Information status distinctions and referring expressions: An empirical study of references to people in news summaries. *Computational Linguistics*, 37(4):811–842.

Ionut-Teodor Sorodoc, Kristina Gulordava, and Gemma Boleda. 2020. Probing for referential information in language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4177–4189, Online. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

A Further details on the `XGBoost` models

As mentioned earlier, for the RFS task, we firstly created the `XGBoost` models using a wide selection of features. Afterwards, we ran a variable importance analysis on the models, and chose a smaller subset of features for each classifier. The selected features are presented in Table 5.

B Further results of importance analysis

Figure 3 depicts the variable importance results for the 2-way and 3-way classification tasks. As mentioned in the paper, there is a high degree of agreement between the ordering of the variables in the 3 models.

To get a better idea about the contribution of each variable to the decisions made by the models, Figure 4 demonstrates the shapley values for 100 random orderings of explanatory variables in the 4-way classification model. The figure clearly shows that the model has failed to learn the `demonstrative` class. For other decisions, the model majorly uses 2 features, namely `DisStat` (referential status) and `Syn` (syntactic role).

Feature	Definition	2-way	3-way	4-way
Syn	Description is provided in the main text.	✓	✓	✓
Entity	Values: Person, Organisation, Location, Number, Other	✓	✓	✓
Gender	Values: male/female/other	✓	✓	✓
DisStat	Description is provided in the main text.	✓	✓	✓
SenStat	Description is provided in the main text.	-	✓	✓
DistAnt_S	Description is provided in the main text (DistAnt).	✓	✓	✓
DistAnt_W	Distance in number of words (5 quantiles)	✓	-	✓
Sent_1	Does RE appear in the first sentence?	✓	✓	✓
MetaPro	Description is provided in the main text.	✓	✓	✓
GloPro	Description is provided in the main text.	✓	✓	✓

Table 5: Features used in the XGBoost models.

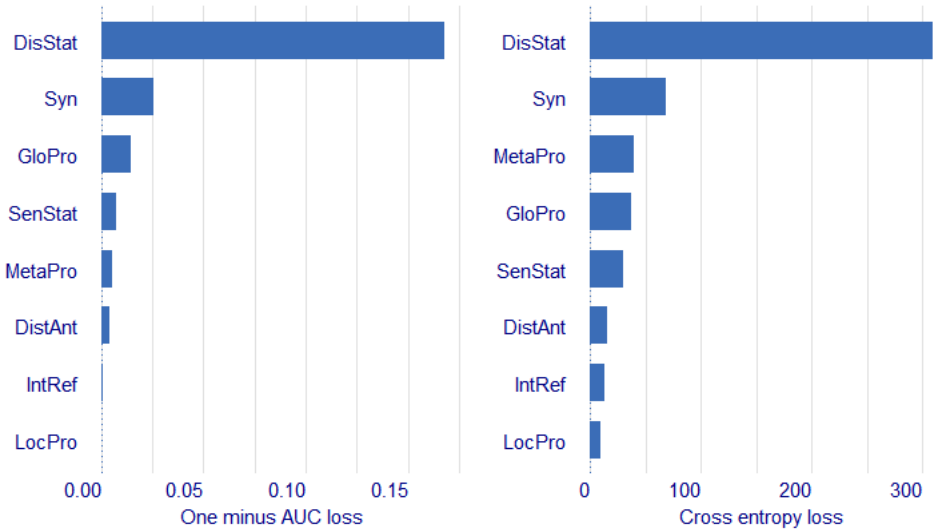


Figure 3: Feature Importance of the XGBoost 2-way (left figure) and 3-way (right figure) predictions.

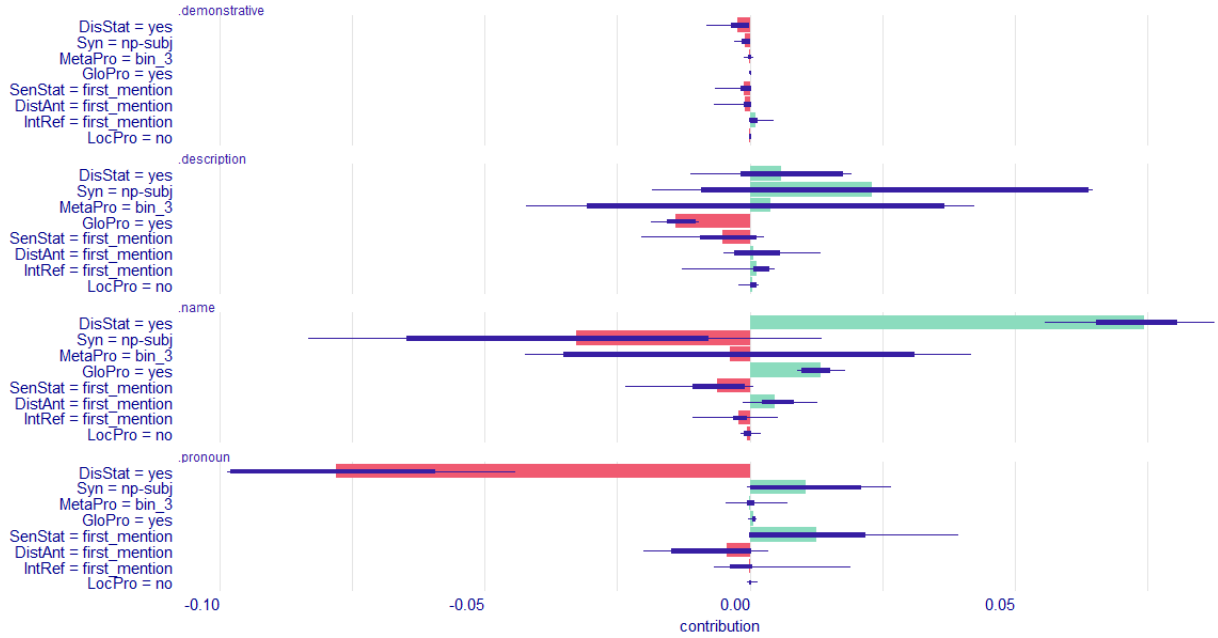


Figure 4: Shapley values with box plots for 100 random orderings of explanatory variables in the XGBoost 4-class model.

HI-CMLM: Improve CMLM with Hybrid Decoder Input

Minghan Wang[1], **Jiaxin Guo**[1], **Yuxia Wang**[2], **Yimeng Chen**[1], **Chang Su**[1], **Daimeng Wei**[1], **Min Zhang**[1], **Shimin Tao**[1], **Hao Yang**[1]
[1]Huawei Translation Services Center, Beijing, China
[2]University of Melbourne, Melbourne, Australia
{wangminghan,guojiaxin1,chenyimeng,suchang8,
weidaimeng,zhangmin186,taoshimin,yanghao30}@huawei.com
yuxiaw@student.unimelb.edu.au

Abstract

Mask-predict CMLM (Ghazvininejad et al., 2019) has achieved stunning performance among non-autoregressive NMT models, but we find that the mechanism of predicting all of the target words only depending on the hidden state of [MASK] is not effective and efficient in initial iterations of refinement, resulting in ungrammatical repetitions and slow convergence. In this work, we mitigate this problem by combining copied source with embeddings of [MASK] in decoder. Notably. it's not a straightforward copying that is shown to be useless, but a novel heuristic hybrid strategy — fence-mask. Experimental results show that it gains consistent boosts on both WMT14 En↔De and WMT16 En↔Ro corpus by 0.5 BLEU on average, and 1 BLEU for less-informative short sentences. This reveals that incorporating additional information by proper strategies is beneficial to improve CMLM, particularly translation quality of short texts and speeding up early-stage convergence.

1 Introduction

In neural machine translation (NMT), autoregressive models decode tokens one-by-one: $p(Y|X) = \prod_i^T p(y_i|y_{<i}|X)$, which ensures the robustness of intrinsic language models but slows down the inference. Non-autoregressive models break the dependency between adjacent tokens: $p(Y|X) = \prod_i^T p(y_i|X)$, enabling to generate all outputs in parallel.

Recent years have witnessed impressive advances in non-autoregressive models, such as fully-NAT and its variants (Gu et al., 2018; Guo et al., 2019; Wang et al., 2019), insertion-based models (Stern et al., 2019; Gu et al., 2019) and iterative refinement models (Lee et al., 2018; Ghazvininejad et al., 2019). Mask-predict CMLM (CMLM) stands out of them owing to both significantly-fast inference and remarkable performance (Ghazvininejad et al., 2019). It extends the masked language model (Devlin et al., 2019) and enables it to solving generation tasks with iterative refinement. In each step, the model decodes target conditioned on m well-predicted tokens with high confidence and $(L-m)\times$[MASK], where L is the length of target (see Section 2 for details). This mechanism leads to the issue that it's liable to generate repeated tokens and slow down the convergence in early-stage iterations. We speculate this is because the proportion of useful tokens, i.e. $m \to 0$, is too small to provide enough information for the next step prediction. Intuitively, the model tends to predict similar or even identical tokens when observing [MASK] only and constantly.

To alleviate this problem, we ameliorate CMLM by incorporating additional information from source embedding into the decoder input (HI-CMLM in short). Experimental results show that it gains consistent boosts on both WMT14 En↔De and WMT16 En↔Ro corpus by 0.5 BLEU on average, and 1 BLEU for less-informative short sentences. This reveals that incorporating additional information by proper strategies is beneficial to improving CMLM, particularly translation quality of short texts and speeding up the convergence of the first four iterations, compared with CMLM.

2 Conditional Masked Language Models

2.1 Model

The architecture of CMLM is a standard encoder-decoder Transformer (Vaswani et al., 2017) without the decoder self-attention mask because the dependency on left tokens has been removed. Formally, given source/target pair (X, Y), the model first predicts the target length based on X before decoding, with objective function:

$$\mathcal{L}_{\text{LEN}} = logP(L|X;\theta). \quad (1)$$

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 167–171,
Aberdeen, Scotland, UK, 20-24 September 2021.

In token prediction at step t, the model refines unobserved tokens $Y^{(t)}_{\text{mask}}$ by minimizing MLM loss:

$$\mathcal{L}_{\text{MLM}} = \sum_{y_i \in Y^{(t)}_{\text{mask}}} logP(y_i|X, Y^{(t)}_{\text{obs}}; \theta). \quad (2)$$

based on a sequence consisting of observed tokens $Y^{(t)}_{\text{obs}}$ and masked tokens $Y^{(t)}_{\text{mask}}$. The total loss function is the sum of length loss and MLM loss:

$$\mathcal{L} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{LEN}}. \quad (3)$$

2.2 Mask-predict Decoding

The decoder runs a *mask* operation, followed by *predict* for T iterations. In each iteration t, it masks the k tokens with the lowest probability scores, where k is determined by a linear decay function of t: $k = L \times \frac{T-t}{T}$. Observed tokens $Y^{(t+1)}_{\text{obs}}$ and masked tokens $Y^{(t+1)}_{\text{mask}}$ are updated by:

$$Y^{(t+1)}_{\text{mask}} = \arg\min(p^{(t)}_i, k) \quad (4)$$

$$Y^{(t+1)}_{\text{obs}} = Y^{(t)} \setminus Y^{(t+1)}_{\text{mask}}, \quad (5)$$

where $p^{(t)}_i$ is the probability score when the model predicts token y_i at step t:

$$y^{(t)}_i = \arg\max P(y_i = w|X, Y^{(t)}_{\text{obs}}) \quad (6)$$

$$p^{(t)}_i = \max P(y_i = w|X, Y^{(t)}_{\text{obs}}), \quad (7)$$

2.3 Training Strategy

To simulate the decoding process in each step, the ground truth target is corrupted by randomly replacing several tokens with [MASK]. The number and the position of the [MASK] follows the uniform distribution so that every token has equal chance to be masked. Then, the model has to recover the corrupted sequence.

2.4 Rethinking Effectiveness of Mask

In *mask-predict*, what should be highlighted is that for the first iteration: $t = 0 \rightarrow k = L$, the model masks all the tokens, thus it predicts entire target sequence merely depending on a full sequence of [MASK] of length L. This leads to the fact that the decoder always requires more than 5 refinement iterations to converge, which is significantly against the original intention to be faster.

We speculate this may result from following reasons: 1) The proportion of Y_{obs} is too small to support the masked language model generating fluency sentences and 2) The representation of Y_{mask} is less informative and distinguishable, and the consecutive [MASK] padding form exacerbates the situation because of lacking useful information inferred from surrounding tokens. We hypothesize that proper initialization of $Y^{(t)}_{\text{obs}}(t \leq 3)$ may be beneficial to speeding up refinement, and improving the final performance. But the question is *what initialization would be helpful?* Put differently, *how to incorporate additional information to Y_{mask} to ameliorate prediction in initial steps.*

3 Method

In this section, we propose three hybrid approaches to incorporate source embeddings and describe modifications on training strategy accordingly.

3.1 Copy of Source Embedding

The most straightforward method is to mix mask tokens with the source embedding. To address the inconsistency of length, we follow the prior work by uniform copy or soft copy (Gu et al., 2018; Lee et al., 2018; Guo et al., 2019; Wang et al., 2019) but with a modified copy function which copies tokens according to their relative position instead of the absolute position. We denote the copy function as $z_i = \Phi(\mathbf{e})$:

$$d_{ij} = -|\frac{i}{L_Y} - \frac{j}{L_X}| \quad (8)$$

$$\alpha_{ij} = \frac{\exp(d_{ij})/\tau}{\sum_j^{L_X} \exp(d_{ij})/\tau} \quad (9)$$

$$z_i = \sum_j^{L_X} \alpha_{ij} \cdot e_j, \quad (10)$$

where d_{ij} is the distance between the target token y_i and source token x_j normalized by specific length, α_{ij} represents the weight and e_j is the embedding of source x_j. τ is the temperature of the softmax function set as 0.2 in our experiment.

3.2 Hybrid Strategy

We compare three strategies to mix copied source embeddings with masks (denoted as Z and M for simplicity) with the baseline (All mask) — All copied, Weighted Sum and our heuristically-proposed Fence Replace, as shown in Figure 1.

All Mask: It's exactly same as CMLM, served as baseline.

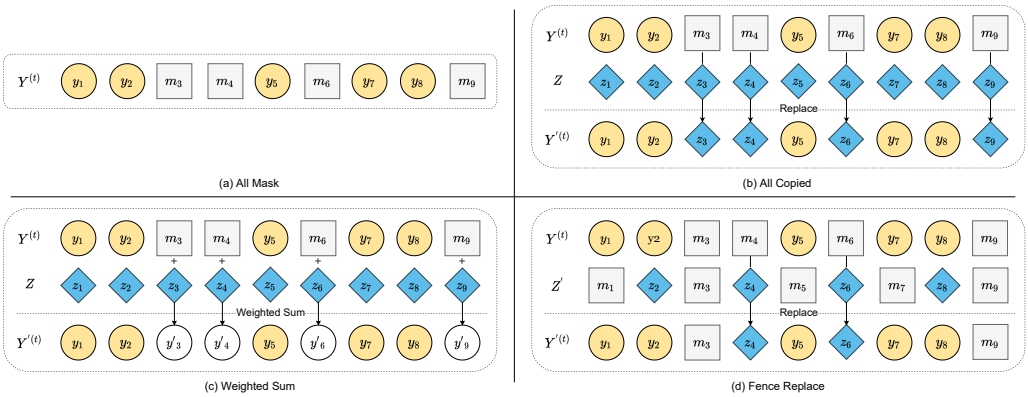


Figure 1: The hybrid strategy.

All Copied: We replace all embeddings of mask with copied source embedding added with position embedding, which is equivalent to entirely using the information from source.

Weighted Sum: It mixes the information by adding M and Z elementwise with a certain weight. We test 4 combinations ranging from 0.2 to 0.8 for M and Z with the stride set as 0.2, i.e. $(0.2, 0.4, 0.6, 0.8)M + (0.8, 0.6, 0.4, 0.2)Z$ and report the best result where the weights are 0.6 and 0.4 for M and Z, respectively.

Fence Replace: During experiments, we find it's important to dynamically change the volume of information added to the decoder input with addition of Y_{obs}. Concretely, Z may become noise instead of useful signals when Y_{obs} has been fully capable to support MLM independently.

Therefore, we propose to replace the masked tokens at odd positions with half of Z exploited, avoiding the decoder input to incorporate too much information and ultimately act as noise. More formally, we first define a mask $(0, 1, 0, 1, ..., 0, 1)$ like a fence with length of L and apply it to stagger the M and Z into a mixed embedding Z', where odd positions are filled with Z and even positions are filled with M. Finally, we replace the embedding of Y_{mask} with the mixed embedding of specific positions.

3.3 Training

To fit the proposed method and meanwhile take full advantage of the masked language model, we modify the training strategy by randomly replacing the subset of the original masked token with the copied source embedding, so that the proportion of corrupted tokens can be unchanged. We apply this method to train the model under all hybrid strategies, including All Mask, for convenient comparison, so it differs from the original CMLM in training.

4 Experiments

4.1 Experimental Setup

We evaluate HI-CMLMs with the proposed hybrid strategies on standard machine translation benchmarks including WMT14 En↔De and WMT16 En↔Ro in both directions.

Datasets The sizes of the dataset are 4.5M and 610k for En↔De and En↔Ro respectively. We create the knowledge distilled data as suggested in (Gu et al., 2018; Zhou et al., 2020) with same configurations. BPE (Sennrich et al., 2016) is used for tokenization with the vocabulary size set to 42k and 40k for En↔De and En↔Ro.

Model Configurations We apply the same weight initialization method and configurations on hyperparameters as prior work: n_{layers} = 12, n_{heads} = 8, d_{hidden} = 512, d_{FFN} = 2048 (Ghazvininejad et al., 2019; Vaswani et al., 2017). Our model is trained on 4 Tesla V100 GPUs with the max batch size of 8k tokens per card. Adam (Kingma and Ba, 2015) is used for optimization. The learning rate warms-up for 20k steps to 5e-4 and decays with the inversed-sqrt schedular. We implement models in the experiment with fairseq (Ott et al., 2019).

4.2 Results and Analysis

Table 1 shows the performance of the proposed HI-CMLM with the BLEU score (Papineni et al., 2002). For each language pair, the model obtains consistent improvements with the Fence Replace, but no gains with another two.

Model	En-De	De-En	En-Ro	Ro-En
Transformer (Vaswani et al., 2017)	27.30	-	-	-
Transformer (Our Implementation)	27.72	32.04	34.03	33.93
CMLM (Ghazvininejad et al., 2019)	27.03	30.53	33.08	33.31
CMLM (Our Implementation)	26.89	30.71	32.94	33.07
HI-CMLM + All Mask	27.01	30.74	32.89	33.03
HI-CMLM + All Copied	26.76	30.82	32.74	32.95
HI-CMLM + Weighted Sum	26.81	30.79	32.80	33.14
HI-CMLM + Fence Replace	**27.42 (+0.53)**	**31.32 (+0.61)**	**33.36 (+0.42)**	**33.51 (+0.44)**

Table 1: The performance of the AT teacher, the baseline CMLM, and the HI-CMLM with different hybrid strategies.

Length	CMLM	HI-CMLM (Fence Replace)
Overall	26.89	27.42 (+0.53)
[0,10)	22.24	**23.27 (+1.03)**
[10,23)	26.46	26.98 (+0.52)
[23,+∞)	27.61	27.81 (+0.20)

Table 2: BLEU scores of target sentences with different lengths at the 10-th step.

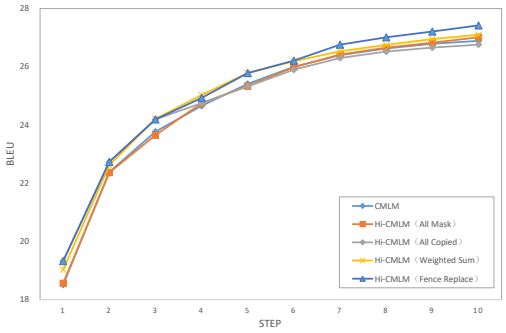


Figure 2: BLEU scores of every step with max_iter=10 for all hybrid strategies as well as the baseline.

To further investigate why Fence Replace stands out, we draw outputs of each step for four strategies in Figure 2 with max_iter=10. It shows from step 4, the model with All Copied and Weighted Sum strategy start to fall back to the All Mask level, which means for the later steps, the added information turns into noise, but can be appropriately controlled by the Fence Replace. We empirically explain how it controls below.

Results on different length targets We evaluate performance of Fence Replace over three bins based on the length of targets: $[0, 10)$, $[10, 23)$, and $[23, \infty)$. Table 2 shows more gains are obtained on short sentences. Intuitively, we guess the benefits result from the enhanced condition of $p(y_i|X)$, by complementing sparse X of short sentences with informative $mix(Z, M)$ in early steps. But if so,

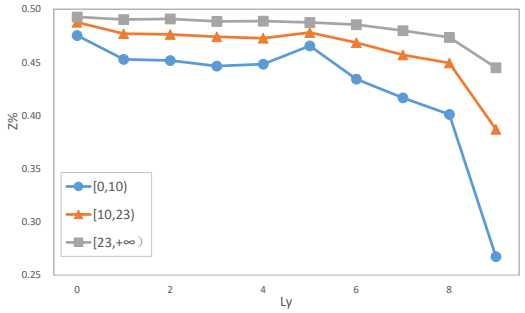


Figure 3: The proportion of copied source embedding within the masked area for sentences with different length when applying the Fence Replace strategy.

why All Copied and Weighted Sum do not work?

We show it's not the whole story. In Figure 3, the proportion of Z actually used for replacement has been reduced from step 6 for all length bins due to the sparsity of re-masked tokens, particularly for shorter sentences, it's much less than 50% that is pre-determined by fence and dropped faster. So the outstanding performance of Fence Replace is not only attributed to incorporated source embedding but the significantly-reduced proportion of Z in later steps as well, effectively avoiding Z from acting as noise.

This comprehensively reveals that the Fence Replace can flexibly balance the information feed to decoder inputs, more signals in early-stage refinement and less information in later steps.

5 Conclusion

We present HI-CMLM, an extension of CMLM by mixing source embedding with a hybrid strategy — Fence Replace, which can appropriately balance the information applied to the model. It achieves consistent improvements on two benchmarks in both directions, particularly short sentences.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 6111–6120. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11179–11189.

Junliang Guo, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. 2019. Non-autoregressive neural machine translation with enhanced decoder input. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3723–3730. AAAI Press.

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

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1173–1182. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5976–5985. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 5377–5384. AAAI Press.

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Using BERT for choosing classifiers in Mandarin

Jani J. Järnfors[♠], **Guanyi Chen**[♠], **Kees van Deemter**[♠] and **Rint Sybesma**[♣]
[♠]Utrecht University [♣]Leiden University
j.j.jarnfors@students.uu.nl {g.chen, c.j.vandeemter}@uu.nl
r.p.e.sybesma@hum.leidenuniv.nl

Abstract

Choosing the most suitable classifier in a linguistic context is a well-known problem in the production of Mandarin and many other languages. The present paper proposes a solution based on BERT, compares this solution to previous neural and rule-based models, and argues that the BERT model performs particularly well on those difficult cases where the classifier adds information to the text.

1 Introduction

The grammar of Mandarin and certain other Chinese languages requires that, in a number of syntactic positions, a noun must be preceded by a *classifier* word. Classifiers often give a rough indication of the kind of entity denoted by the noun. For example, the classifier "只" (zhī) in the Noun Phrase (NP) "一只狗" (yì zhī gǒu; *a dog*) indicates the noun "狗" (gǒu; *dog*) is an animal. It is worth noting that, in addition to Mandarin, classifiers also play a critical role in a few other languages, especially the East Asian languages, such as Korean, Japanese, and Vietnamese (Aikhenvald, 2000). Generally speaking, it is, in many ways, not unlike *types* in functional programming languages like Haskell, which add to each function defined by the programmer a broad semantic categorisation of that function (Thompson, 2011).

Mandarin contains a large number of classifiers, and although the choice of classifier is limited by the (head) noun with which the classifier is associated, this may still leave several options, which may sometimes produce a different meaning, e.g.,

(a) 一个 电脑/ 一台 电脑
yí gè diànnǎo / yí tái diànnǎo
'a computer'

(b) 一个 老师/ 一位 老师
yí gè lǎoshī / yí wèi lǎoshī
'a teacher'

(c) 一个 人/ 一群 人
yí gè rén / yí qún rén
'a person / people'

(d) 一杯 咖啡/ 一听 咖啡
yì bēi kāfēi / yì tīng kāfēi
'a cup/can of coffee'

Although each of these cases involves classifier choice, the problem of choosing a classifier is likely to be more challenging in those cases, such as (b)-(d), where the classifier adds information, for example, in terms of politeness ((b), neutral vs. polite), number ((c), singular vs. plural), or quantity ((d), a cup vs. a can of coffee). This is perhaps clearest in the case of (d), where "杯" (bēi; *cup*) and "听" (tīng; *can*) indicate different containers, and consequently different quantities, of coffee; these classifiers are known as measure words, as opposed to the "pure" classifiers of (a)-(c).

Researchers have asked what determines the choice of classifier, constructing algorithms that predict what classifier suits a given discourse context. The most sophisticated model we are aware of is Peinelt et al. (2017). Ambitiously, these authors decided to deal with classifiers of *all* different types, also including measure words for instance, which are difficult to predict because they add information. They approached the problem as follows: Given a sentence in which a classifier is yet to be realised, and the head noun is flagged, predict the missing classifier. For example, in the input:

(1) 一⟨CL⟩ 精彩的⟨h⟩球赛⟨/h⟩
yì ⟨CL⟩ jīngcǎi de ⟨h⟩qiúsài⟨/h⟩
'a wonderful ball game'

⟨CL⟩ indicates where the missing classifier is and the ⟨h⟩ tag pair flags the head noun. The authors construct a large-scale classifier dataset, namely

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 172–176,
Aberdeen, Scotland, UK, 20-24 September 2021.

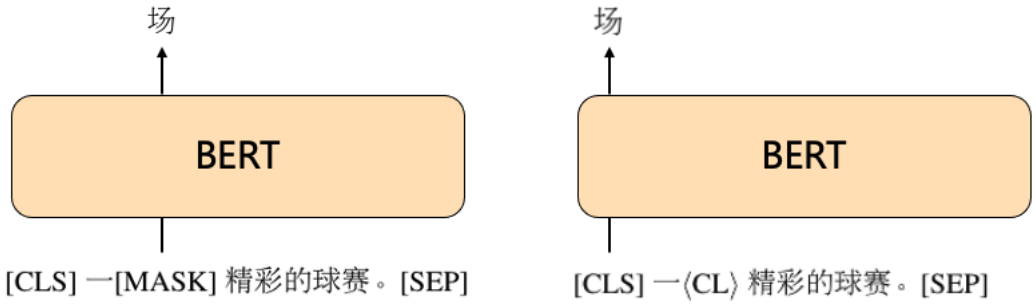


Figure 1: Sketch of our BERT-based Classifier selection models: predicting the classifier by unmasking the [MASK] (left); predicting the classifier as classification (right).

ChineseClassifierDataset[1] (henceforth, CCD) by extracting and filtering data from publicly available Chinese corpora. They did experiments on their CCD corpus with several baselines, including a rule-based system, two machine learning based system, and a LSTM-based system (Hochreiter and Schmidhuber, 1997). An initial evaluation study indicated that the LSTM achieved the best performance.

Our own work takes the same perspective as Peinelt et al. (2017). But although the *performance* of the model of Peinelt et al. is encouraging, it still leaves considerable room for improvement; in particular, the question comes up whether BERT, with its superior ability to take context into account, might perform better. In addition, the model of Peinelt et al. offers only limited *insight*, because it does not distinguish between different types of classifiers. In other words, the performance of the model may mask important differences between different types of classifier choice. A good way to address this limitation would be to make use of an existing categorisation of classifier types. But although linguists generally agree that "true" (or "sortal") classifiers should be distinguished from measure words (Croft, 1994; Cheng and Sybesma, 1999), there exist subtle disagreements regarding exactly how these sub-types should be defined, and what further divisions between sub-types should be taken into account. Subtypes are often described by example, without computationally implementable criteria or explicit lists of classifiers (Zhang, 2013). To our knowledge, Her and Lai (2012) are the only ones to provide comprehensive lists of classifiers of various subtypes, and in what follows we will make use of these lists.

[1] github.com/wuningxi/ChineseClassifierDataset

In Section 2, we introduce two different BERT-based models, one of which uses word masking and one of which performs classification. In Section 3, we report on our comprehensive evaluation experiments, in which we compare our BERT-based models with each other and with several baselines, using the CCD dataset.

2 Choosing Classifiers using BERT

We use BERT to accomplish the task of choosing classifiers in two ways: an unsupervised way (i.e., predicting classifiers by unmasking masked tokens) and a supervised way (i.e., fine-tuning BERT on the task of classifier prediction).

2.1 Unmasking Masked Classifiers

In order to assess how well BERT, as a masked language model, can model classifiers, we tried to use BERT without any fine-tuning on the task of classifier selection. Specifically, as shown in Figure 1 (left), we replace the classifier indicator ⟨CL⟩ with the [MASK] symbol of BERT and ask BERT to unmask it. [2] The unmasked token serves as the predicted classifier. (Note that addressing the classifier selection task in this way will sometimes produce words that are not classifiers.) We refer to this model as MLM.

2.2 Classifying Classifiers

Additionally, we test BERT in its classic use. To do this, we fine-tune BERT on the CCD as a multi-class classification task, where there are 172 classes (i.e., 172 classifier words) in total, and make a prediction with the help of the [CLS] symbol (see Figure 1 (right)). We refer to this model as BERT.

[2] Since our experiments suggested that the head flag (i.e., ⟨h⟩ and ⟨/h⟩) makes no contribution to classifier selection, we drop it to speed up the prediction.

2.3 Research Questions

At the start of our research, we formulated the following hypotheses and research questions.

1. Since BERT models context closely and is pre-trained on large scale corpora, we expect it to outperform other models;

2. How do the two BERT-based models compare? Although we expect BERT to outperform MLM, we were curious to see how well MLM performs.

3. We are curious how well BERT can handle classifiers that add information (concretely, in this paper: measure words, plurality, and politeness).

3 Experiments

3.1 Setup

Dataset. In total, there are 681,102 sentences in the CCD dataset. We split the dataset into training (60%), development (20%), and test (20%) sets following Peinelt et al. (2017).

Baselines. We tried several baseline models proposed in Peinelt et al. (2017), including: (1) a rule based model (Rule): given a head noun, assign the most frequent classifier associated with it in the training data. If two or more classifiers are equally frequent, one of the classifiers is randomly assigned. If the head noun does not appear in the training data, then the classifier "个" (gè) (which is particularly frequent and often seen as a "default" classifier) is assigned; (2) a LSTM model: for this model, we use a bi-directional LSTM (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) to encode the input; it makes predictions using the hidden representation of the last time step.

Metrics. We evaluate each model in terms of accuracy, macro-averaged precision, recall, and F1. Additionally, since the distribution of the CCD is skewed (e.g., more than 25% of the sentences use "个" (gè)), we also report the weighted averaged precision, recall, and F1.

Implementation Details. For BERT, we use the "bert-base-chinese" version[3]. When fine-tuning, we set the learning rate to 2e-5 and batch size to 150. For the LSTM, we set the batch size to 256, the hidden size to 300, and the learning rate to 2e-5. We use pre-trained Chinese word embeddings from Li et al. (2018)[4].

3.2 Results and Analysis

Table 1 charts the performance of each model. The results confirm the assumption of our first research question that BERT performs the best, defeating all models on all metrics with large margins. For example, for accuracy, compared to the second best model LSTM, BERT boosts performance from 70.44% to 81.71%. Considering its simplicity, the rule-based system achieved a considerably good performance, with higher macro-averaged precision, recall, and F1 than LSTM and with a similar accuracy as MLM. This also confirms the viability of a dictionary-based classifier selector, such as the one embedded in a previous Chinese surface realiser (Chen et al., 2018)).

MLM, as a model without any training on CCD, performs remarkably well. It receives the second best weighted average as well as micro-averaged F1 (in line with our second research question). Note that, as was mentioned, there is no guarantee that the outputs of MLM are classifiers. Concretely, during testing, MLM produces 1566 word types that are not classifiers. This is one of the reasons why its fine-tuned version, BERT, has a major improvement on the (macro-averaged and weighted averaged) recall scores as well as the accuracy. Nonetheless, it surprised us that MLM can produce a greater *variety* of classifiers than all other models. More specifically, out of 172 classifiers available in CCD, MLM has correctly produced 160 different classifiers, comparing to the 140 of Rule, 108 of LSTM, and 136 of BERT. This suggests MLM can sometimes handle rarely seen classifiers.

Regarding the last research question, we looked into measure words, plurality, and politeness respectively. First, we categorise classifiers in CCD into three sub-categories: true classifiers, measure words, and dual classifiers (i.e., classifiers that can function either as true classifiers or as measure words) based on the lists provided by Her and Lai (2012)[5]. Table 2 breaks down the performance into different sub-types of classifiers. As we can see,

[3] huggingface.co/bert-base-chinese

[4] These are word embeddings trained by skip-gram on 9 large Chinese corpora with 300 dimensions. It is available at: github.com/Embedding/Chinese-Word-Vectors

[5] These classifier lists were constructed on the basis of the Mandarin Daily Dictionary of Chinese Classifiers (MDDCC).

Model	Accuracy	Macro-averaged			Weighted-averaged		
		Precision	Recall	F1	Precision	Recall	F1
Rule	61.89	34.87	20.50	23.39	58.23	61.90	58.24
LSTM	70.44	33.11	20.12	22.48	67.90	70.44	68.12
MLM	62.22	51.91	33.40	37.68	77.28	62.23	68.21
BERT	**81.71**	**52.86**	**38.10**	**40.77**	**80.70**	**81.71**	**80.77**

Table 1: Evaluation Results of each model on CCD. The best results are **boldfaced**, whereas the second best are underlined. MLM is the model that uses BERT as a masked language model, while BERT is the fine-tuned BERT.

Category	Frequency	Accuracy
True Classifier	85,917	87.8
Dual Classifier	10,817	65.2
Measure Words	11,317	61.1

Table 2: BERT's performance on different types of classifiers; frequency of each type in the CCD test set.

although measure words appear more frequently in CCD than dual classifiers, they still receive a significantly lower accuracy.

Second, for politeness, the only frequent enough[6] politeness classifier is "位" (wèi), which expresses politeness when referring to a person. "位" appears 6737 times in the training data, but only obtains a recall score of 59.87%, which is low compared to equally frequent classifiers (classifiers with frequencies in the range of [5000, 8000) have a average recall score of 77.84%). The confusion matrix[7], shows that it is highly likely to be confused with its neutral alternative "个" (gè).

Third, regarding plurality, we pick out frequent-enough classifiers that only convey the meaning of plurality[8], including "群" (qún), "堆" (duī), "些" (xiē), "套" (tào), "对" (duì), and "双" (shuāng). Their recall scores are 52.51% (2453), 52.12% (1914), 56.51% (1910), 34.57% (1308), 62.39% (1321), and 76.49% (806), respectively, where the number in brackets is the frequency of that classifier in the training set. Meanwhile, the average recall of the range [800, 1500) and [1500, 3000) are 61.48% and 76.97%. It is interesting that BERT does a relatively good job for handling plural classifiers meaning "pair" (i.e., "对", and "双") while failing to handle plural classifiers meaning "multiple" (i.e., "群", "堆", "些", and "套"). All in all, classifiers that add information regarding measurement, plurality and politeness could not be properly selected. One explanation is that their context cannot provide enough information to pick the right classifier. Thus, for the last research question, BERT does not work well on handling classifiers that add information.

Distance between the Classifier and the Head Noun. We also explore factors that might influence the decisions of BERT. First, we consider the *distance* between the classifier and the head noun. For instance, for example (1), there is a pre-modifier consisting of two words between the classifier "场" (chǎng) and the head noun "球赛" (qiúsài; *football match*). Thus, the distance for example (1) is 2. We expect that the larger the distance is, the worse BERT performs. In our experiments, for correct predictions, the average distance (in terms of the number of words) is 1.04 while for incorrect predictions it is 1.15. An un-paired t-test confirms that distance has a negative effect on the model's performance ($p < .001$).

4 Discussion

We conclude that (1) contextualised pre-trained models (i.e., BERT and MLM) perform remarkably well on the task of choosing classifiers in Mandarin, and fine-tuning helps improve the recall of choosing classifiers; (2) a simple rule-based system has respectable performance; (3) in terms of accuracy, a pre-trained masked language model (i.e., MLM) was able to select proper classifiers about equally well as the above rule-based system; (4) BERT struggles to predict classifiers that add information (measurement, plurality, politeness).

The last finding confirms our (linguistically well-

[6]We define a classifier is *frequent enough* if it appears more than 50 times in the training set.

[7]The full confusion matrix is too large to print here but, together with the system outputs, is available at: github.com/a-quei/bert-chinese-classifier

[8]Some classifiers have multiple meanings, one of which expresses plurality.

established) expectation that some classifier occurrences cannot be predicted from their linguistic context alone since they themselves carry additional information. Since the choice of classifier is not deterministic (e.g., consider the choice between "个" and "台" in example (a)), the type of corpus evaluation that was performed in this paper arguably does not "tell the whole story" regarding the quality of the different models. To remedy this issue, we plan two further experiments, each of which starts from the observation that the classifier that was chosen in a given linguistic context in the corpus will often not be the only felicitous choice.

One experiment will focus on *speakers*. We will ask several participants to choose classifiers given a linguistic context. By comparing the outcomes of such an elicitation experiment with the CCD corpus, we will obtain a better understanding of the variations that exist between speakers and of the difficulty of the task that we have set our algorithms. By thus asking multiple participants to accomplish the same task as our algorithms, we will obtain a new corpus, in which each linguistic context is associated with a bag of (1 or more) possible classifiers. This new dataset will enable us to conduct a new, non-deterministic evaluation of the models.

Another additional experiment will have human *readers* judge the acceptability of each classifier choice that is made by a given model. Reader experiments of this kind are a standard tool in judging the quality of decisions taken by a natural language generation algorithm (e.g. van der Lee et al. (2019)) and will give rise to a new set of analyses analogous to the ones in the present paper, which will give us a better understanding of the quality of the decisions that are taken by each model.

In the future, we also plan to extend the models we tested in this study. For example, regarding the pre-trained language model, a promising candidate to investigate is ERNIE (Sun et al., 2020), which has proved to be more powerful in modelling Mandarin Chinese. Regarding the unsupervised MLM setting, the following option would be worth trying: instead of choosing the most probable word type from the whole vocabulary, one could ask the model to output the most probable classifier from all classifiers.

References

Alexandra Y. Aikhenvald. 2000. Classifiers. a typology of noun categorization devices. *NewYork: Oxford University Press*.

Guanyi Chen, Kees van Deemter, and Chenghua Lin. 2018. SimpleNLG-ZH: a linguistic realisation engine for Mandarin. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 57–66, Tilburg University, The Netherlands. Association for Computational Linguistics.

Lisa Lai-Shen Cheng and Rint Sybesma. 1999. Bare and not-so-bare nouns and the structure of np. *Linguistic inquiry*, 30(4):509–542.

William Croft. 1994. Semantic universals in classifier systems. *Word*, 45(2):145–171.

One-Soon Her and Wan-Jun Lai. 2012. Classifiers: The many ways to profile'one'—a case study of taiwan mandarin. *International Journal of Computer Processing Of Languages*, 24(01):79–94.

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

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368, Tokyo, Japan. Association for Computational Linguistics.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on Chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 138–143, Melbourne, Australia. Association for Computational Linguistics.

Nicole Peinelt, Maria Liakata, and Shu-Kai Hsieh. 2017. ClassifierGuesser: A context-based classifier prediction system for Chinese language learners. In *Proceedings of the IJCNLP 2017, System Demonstrations*, pages 41–44, Tapei, Taiwan. Association for Computational Linguistics.

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

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975.

Simon Thompson. 2011. *Haskell: the craft of functional programming*, third edition. Addison-Wesley. More information at www.haskellcraft.com.

Niina Ning Zhang. 2013. *Classifier Structures in Mandarin Chinese*, volume 263. Walter de Gruyter.

Enriching the E2E dataset

Thiago Castro Ferreira
aiXplain, inc.
Federal University of Minas Gerais
thiagocf05@ufmg.br

Helena Vaz
Federal University of Minas Gerais
Belo Horizonte, Brazil
lenavaz13@gmail.com

Brian Davis
ADAPT Research Centre
Dublin City University
brian.davis@adaptcentre.ie

Adriana Pagano
Federal University of Minas Gerais
Belo Horizonte, Brazil
apagano@ufmg.br

Abstract

This study introduces an enriched version of the E2E dataset, one of the most popular language resources for data-to-text NLG. We extract intermediate representations for popular pipeline tasks such as discourse ordering, text structuring, lexicalization and referring expression generation, enabling researchers to rapidly develop and evaluate their data-to-text pipeline systems. The intermediate representations are extracted by aligning non-linguistic and text representations through a process called *delexicalization*, which consists in replacing input referring expressions to entities/attributes with placeholders. The enriched dataset is publicly available.[1]

1 Introduction

Data-to-text NLG is the computational task which aims to generate text from non-linguistic data (Reiter and Dale, 2000; Gatt and Krahmer, 2018). Applications of this task have become increasingly common, such as RDF-to-text (Castro Ferreira et al., 2020), AMR-to-text (Ribeiro et al., 2019), dialogue response generation (Dušek et al., 2018), robot-journalism (Rosa Teixeira et al., 2020), etc.

The growth of the field can be partially explained by increasing availability of focused data-to-text resources, such as WebNLG (Gardent et al., 2017b,a), E2E (Novikova et al., 2017; Dušek et al., 2018), ROTOWIRE (Wiseman et al., 2017), GenWiki (Jin et al., 2020) and KELM (Agarwal et al., 2021).

As with other automatic text generation fields, such as Machine Translation, significant advances in deep learning (Cho et al., 2014; Sutskever et al., 2014), along with an increasing number of data-to-text resources, have resulted in upsurge in neural end-to-end applications targeted towards data-to-text NLGk (Gardent and Narayan, 2018). Hence, given a corpus consisting of pairs between a meaning representation (MR) and its corresponding textual verbalization, a deep learning approach is usually trained in an end-to-end style, learning implicit parameters to convert the input MR into textual output. Although these approaches have shown to generate more fluent output, they also pose problems and challenges, in particular with respect to the semantic *adequacy* and overall faithfulness of the text (Wang et al., 2020). For example, some studies have shown that neural end-to-end data-to-text approaches may *hallucinate* (Rohrbach et al., 2018; Wang et al., 2018), i.e. adding information in the text which are not contained in the input data or which are untrue. This is not a trivial issue, given that accuracy of the generated output is in general considered more important than its fluency (Reiter and Belz, 2009). More importantly poor semantic adequacy is in particular unacceptable for practical applications (Dale, 2020). Furthermore, Castro Ferreira et al. (2019) has shown that traditional pipeline data-to-text systems (Reiter and Dale, 2000), which generate text from data in several explicit intermediate steps, may generalize better to new domains and in turn generate more semantically adequate text than end-to-end approaches in the context of the WebNLG corpus.

Although the current data-resources have benefited the development of end-to-end neural models, the same can not be said for pipeline systems, since these resources usually only consist of raw meaning representations and their final verbalizations. Aiming to decrease data sparsity and make data-to-text models more generalizable and generate more adequate texts, many approaches aim to extract alignments between the non-linguistic and text representations, and then use these alignments to build explicit intermediate representations for a more controllable generation process (Juraska et al., 2018; Xu et al., 2021). As examples, all

[1] https://github.com/ThiagoCF05/EnrichedE2E

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 177–183,
Aberdeen, Scotland, UK, 20-24 September 2021.

the data-driven participating models of the E2E work by first converting the meaning representation into an intermediate template which is later realized into the final text. This is also the case in the WebNLG challenge, which makes use of the eponymous dataset.

In order to make it easy for researchers to rapidly develop and evaluate data-to-text pipeline systems, Castro Ferreira et al. (2018b) enriched the WebNLG corpus, one of the most popular data-to-text resources. The study extracts intermediate representations for popular pipeline tasks such as discourse ordering, text structuring, lexicalization and referring expression generation. Intermediate representations are automatically extracted by aligning the non-linguistic and text representations through a process called *delexicalization*, which consists of replacing in the texts referring expressions to input entities/attributes with placeholders. The same extraction process with respect intermediate representations above is applied to the recent CACAPO dataset, which is both multilingual (Dutch and English) and multi-domain, containing up to 10,000 sentences (van der Lee et al., 2020).

Highly inspired by the work of Castro Ferreira et al. (2018b) and van der Lee et al. (2020), our study aims to delexicalize and provide pipeline intermediate representations for another very popular data-to-text dataset: the E2E dataset. We believe that the enriched version of the E2E will provide another data-resource so researchers can better investigate data-driven pipeline systems, their subtasks as well as its comparison with state-of-the-art end-to-end systems.

2 The E2E Dataset

The E2E dataset is a resource initially constructed for training end-to-end, data-to-text applications in the restaurant domain. It consists of 50,602 English verbalizations to 5,751 dialog-act-based meaning representations (Novikova et al., 2017). The dataset is split into training, validation and test sets in a ratio of 76.5%, 8.5% and 15%, respectively.

An example of a pair between a meaning representation (top) and its corresponding text (middle) is depicted in Figure 1. Each meaning representation consists of 3-8 attribute-value pairs, picked from a list of 8 attributes depicted in Table 1. Verbalizations were collected through crowd-sourcing using pictures as stimuli. According to the creators, representing the inputs visually allowed the collection of more natural and informative human references phrases than depicting meaning representations (Dušek et al., 2018).

Attribute	Example Values
name	*The Punter*, *The Waterman*, ...
eatType	restaurant, pub
familyFriendly	yes / no
priceRange	cheap, high, moderate ...
food	Indian, Japanese, Chinese ...
near	*Café Rouge*, ...
area	city center, riverside ...
customerRating	low, average, high ...

Table 1: Attributes contained in a meaning representation of E2E and examples of values.

Although the crowd-workers were asked to verbalize all the information contained in the meaning representation, the creators of the corpus decided to do not penalize those who skipped some information. For this reason, the corpus may also be used to study experiments for the content selection task of pipeline data-to-text systems.

The E2E dataset differs from the WebNLG corpus, which focused on semantic variation, as it leverages higher lexical and syntactical variations, having an average of 8.1 reference verbalizations per meaning representation. The corpus is also bigger than other similar corpora such as SFRest (Wen et al., 2015), a corpus in the domain of Hotels and Restaurants with 5,192 verbalizations to 1,950 meaning representations; and Bagel (Mairesse et al., 2010), with 404 texts verbalizing 202 meaning representations.

3 Delexicalization

Following the method used by Castro Ferreira et al. (2018b) in the WebNLG corpus, we aimed to decrease the data sparsity of the corpus and to align a meaning representation with its corresponding text by *delexicalizing* the texts. The delexicalization process works by replacing the referring expressions to the values of the attributes for placeholders representing the attributes. Figure 1 shows an example of a meaning representation, the final verbalization and its delexicalized version (bottom).

The process was conducted differently for training and validation/test parts of the corpus as explained in the following sections.

3.1 Training Data

The process of delexicalizing the training data started by string matching the values of the attributes in the text and replacing them for the spe-

Attribute	Value
name	The Wrestlers
eatType	coffee shop
food	Japanese
priceRange	less than £20
area	riverside
familyFriendly	no
near	Raja Indian Cuisine

↓

Near Raja Indian Cuisine in Riverside is The Wrestlers. It is a Japanese restaurant, has reasonable prices but is not kid friendly.

↓

Near _NEAR_ in _AREA_ is _NAME_ . _NAME_ is a _FOOD_ restaurant , has _PRICERANGE_ prices but is _FAMILYFRIENDLY_ .

Figure 1: Example of the attribute-value pairs of a meaning representation (top), its corresponding verbalization (middle) and a delexicalized template annotated in this study (bottom).

cific placeholder of the attribute i.e _NAME_ or _EATTYPE_ etc. All the partial delexicalized templates were then manually reviewed and annotated by students of linguistics.

Students The students of Linguistics were recruited through a call which announce the task offering university credits in exchange. In total, 10 students volunteered to conduct the annotation.

Website In order to facilitate the annotation, the authors created a website, where, for each annotation instance, the annotators were given access to the input meaning representation, the delexicalized meaning representation, the text and the delexicalized text to be reviewed and corrected. Moreover, a checkbox was provided so the annotators could indicate any problem in the data such as wrong information or hallucination, i.e. verbalization of information which is not contained in the meaning representation.

3.2 Validation/Test Data

In order to accelerate the annotation of the validation and test sets of the corpus, we first trained a Named Entity Recognition and Classification (NERC) tool based on BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) using the annotated training data, effectively substituting placeholders for named entities. We then replaced the referring expressions which weren't recognized by the NERC model by string matching (and substituting) the attribute values within the text. Finally, to assure the quality of the data, especially the test set, the authors manually reviewed each instance of both parts of the data.

NERC Settings Our NERC model consists of the base, cased version (`bert-base-cased`) of an English BERT encoder (Devlin et al., 2019) based on the Transformer architecture (Vaswani et al., 2017) with 12 layers, hidden dimensions of 768, 12 heads and 109M parameters in total. On top of BERT, the model has a classifier consisting of a projection layer with the `Mish` activation function (Misra, 2020) and a softmax layer. The model was trained in the train split of the enriched corpus for 20 epochs with early stop of 3 in an annotated subset of the dev split. Learning rate and batch size were set to 1e-5 and 64, respectively.

Given a text to be delexicalized, the model works by first tokenizing it and encoding the tokens in their context-sensitive embedding representations. These embeddings are then fed into the classifier head, which classifies each token. In order to know whether each token is contained within a mention of one of the 8 attributes and where each of these mentions starts and ends in terms of tokens, we used the `IOB2` format, popular in NERC applications (Ramshaw and Marcus, 1995). In total, the model classifies each token according to 17 classes[2], one that indicates whether a token is not a mention and 2 for each one of the attributes, pointing whether a mention starts (`B-`) and the remaining tokens of the mention (`I-`).

4 Explicit Intermediate Representations

Based on the alignments between the meaning representation and the text provided by the delexi-

[2] `O`, `B-FOOD`, `I-FOOD`, `B-NAME`, `I-NAME`, `B-EATTYPE`, `I-EATTYPE`, `B-FAMILYFRIENDLY`, `I-FAMILYFRIENDLY`, `B-AREA`, `I-AREA`, `B-CUSTOMERRATING`, `I-CUSTOMERRATING`, `B-PRICERANGE`, `I-PRICERANGE`, `B-NEAR` and `I-NEAR`

calization process, we can extract several explicit intermediate representations that can help to study several generation phenomenon as well as to build traditional pipeline (rule based or data driven) data-to-text systems, which may generate more adequate texts and to generalize better for new domains (Castro Ferreira et al., 2019).

Similar to Castro Ferreira et al. (2018b), we have enriched the E2E dataset with several intermediate representations about content selection, discourse ordering, text structuring, lexicalization and referring expression generation. These intermediate representations could be used to study each phenomenon as well as to develop a data-driven, pipeline data-to-text system as envisaged by Castro Ferreira et al. (2019).

Content Selection is the task of deciding which information should be verbalized. By comparing the attributes contained in a meaning representation and the presence or absence of their placeholders in the delexicalized template, we are able to automatically extract all the input content for a given verbalisation. In the example of Figure 1 for instance, we can see that the placeholder of the attribute `eatType` (e.g. _EATTYPE_) is not present in the delexicalized template, indicating that it was not selected to be verbalized in the text.

Discourse Ordering is the task of sorting the selected content in the order it should be verbalized. By looking at the order of the placeholders in the delexicalized text, we can extract this order. In Figure 1, looking at the order of the placeholders in the delexicalized template, we see that the sorted list of attributes is: `near`, `area`, `name`, `food`, `priceRange` and `familyFriendly`,

Text Structuring is the task within pipeline data-to-text systems responsible for structuring the outputs of content selection and discourse ordering into paragraphs and sentences. Using Stanza (Qi et al., 2020), we tokenized the sentences of each delexicalized template and considering their placeholders, extracted the sentence plan for each one the attributes verbalized. In Figure 1 for instance, `near`, `area`, `name` were verbalized in the first sentence, whereas `food`, `priceRange` and `familyFriendly` in the second.

Lexicalization aims to find the proper phrases and words to express the content to be included in each sentence. To obtain lexicalization templates similar to the ones used for the neural pipeline system of Castro Ferreira et al. (2019), we again used Stanza in the delexicalized templates to lemmatize determiners and verbs and extract their correct morphological inflection information. Then in these templates, determiners and verbs were replaced by their morphological inflection information and lemmas. For instance, for the delexicalized template depicted in Figure 1, the lexicalization template would be:

> Near _NEAR_ in _AREA_ VP[*Mood=Ind, Number=Sing, Person=3, Tense=Pres, VerbForm=Fin*] be _NAME_ .
> _NAME_ VP[*Mood=Ind, Number=Sing, Person=3, Tense=Pres, VerbForm=Fin*] be DT[*Definite=Ind, PronType=Art*] a _FOOD_ restaurant , VP[*Mood=Ind, Number=Sing, Person=3, Tense=Pres, VerbForm=Fin*] have _PRICERANGE_ prices but VP[*Mood=Ind, Number=Sing, Person=3, Tense=Pres, VerbForm=Fin*] be _FAMILYFRIENDLY_ .

Referring Expression Generation is the task responsible for generating the references to the entities present in the text (Castro Ferreira et al., 2018a). In our case, these entities are the attributes of the meaning representation. Following Castro Ferreira et al. (2018b), we extract the referring expression to the attributes by overlapping an original text and its delexicalized version. In Figure 1, contains examples of extracted references such as *The Wrestlers* and *It* for the `name` attribute value `The Wrestlers` in the meaning representation.

Surface Realization is responsible for taking the last decisions to convert a non-linguistic data into text. In this case, the correct morphological realisation of determiners and verbs as well as detokenizing the text. For this step in specific, we did not extract any kind of information, but refer to the extensive literature that exists on morphological inflection (McCarthy et al., 2019; Vylomova et al., 2020). These tools can be used to correctly realize our extracted lexicalization templates. Moreover, detokenization is a task already solved with high accuracy.

5 Conclusion

This work introduces the enriched version of the E2E dataset (Novikova et al., 2017; Dušek et al., 2018). Together with the enriched version of WebNLG (Castro Ferreira et al., 2018b) and CACAPO van der Lee et al. (2020), this resource will help researchers to carefully investigate particular pipeline processes in data-to-text systems

in the levels of Macro- (e.g., Content Selection, Discourse Ordering and Text Structuring), Microplanning (e.g., lexicalization, aggregation and referring expression generation) and Surface Realization. In particular, we will be able to better analyse how such subtasks could obtain better performance when developed using a rule-based approach or a specific/multitask data-driven system. Moreover, in future work the community will be able to better compare pipeline and end-to-end data-to-text systems in terms of generalization as well as fluency and adequacy of their generated texts.

Acknowledgments

This research was partially funded by the Brazilian agencies CNPq, CAPES, and FAPEMIG. In particular, the researchers were supported by CNPQ grant No. 310630/2017-7, CAPES Post doctoral grant No. 88887.508597/2020-00, and FAPEMIG grant APQ-01.461-14. It was also funded by the ADAPT Centre for Digital Content Technology (`https://www.adaptcentre.ie`) which is funded under the Science Foundation Ireland Research Centres Programme (Grant 13/RC/2106_P2).

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional WebNLG+ shared task: Overview and evaluation results (WebNLG+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 55–76, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 552–562, Hong Kong, China. Association for Computational Linguistics.

Thiago Castro Ferreira, Diego Moussallem, Ákos Kádár, Sander Wubben, and Emiel Krahmer. 2018a. NeuralREG: An end-to-end approach to referring expression generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1969, Melbourne, Australia. Association for Computational Linguistics.

Thiago Castro Ferreira, Diego Moussallem, Emiel Krahmer, and Sander Wubben. 2018b. Enriching the WebNLG corpus. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 171–176, Tilburg University, The Netherlands. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Robert Dale. 2020. Natural language generation: The commercial state of the art in 2020. *Natural Language Engineering*, 26(4):481–487.

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

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands. Association for Computational Linguistics.

Claire Gardent and Shashi Narayan. 2018. Deep learning approaches to text production. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pages 4–9, New Orleans, Louisiana. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on*

Natural Language Generation, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. GenWiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2398–2409, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker. 2018. A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 152–162, New Orleans, Louisiana. Association for Computational Linguistics.

Chris van der Lee, Chris Emmery, Sander Wubben, and Emiel Krahmer. 2020. The CACAPO dataset: A multilingual, multi-domain dataset for neural pipeline and end-to-end data-to-text generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 68–79, Dublin, Ireland. Association for Computational Linguistics.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1552–1561, Uppsala, Sweden. Association for Computational Linguistics.

Arya D. McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sabrina J. Mielke, Jeffrey Heinz, Ryan Cotterell, and Mans Hulden. 2019. The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244, Florence, Italy. Association for Computational Linguistics.

Diganta Misra. 2020. Mish: A Self Regularized Non-Monotonic Activation Function. In *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY, USA.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2018. Object hallucination in image captioning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4035–4045, Brussels, Belgium. Association for Computational Linguistics.

André Luiz Rosa Teixeira, João Campos, Rossana Cunha, Thiago Castro Ferreira, Adriana Pagano, and Fabio Cozman. 2020. DaMata: A robot-journalist covering the Brazilian Amazon deforestation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 103–106, Dublin, Ireland. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*, Montreal, CA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010.

Ekaterina Vylomova, Jennifer White, Elizabeth Salesky, Sabrina J. Mielke, Shijie Wu, Edoardo Maria Ponti, Rowan Hall Maudslay, Ran

Zmigrod, Josef Valvoda, Svetlana Toldova, Francis Tyers, Elena Klyachko, Ilya Yegorov, Natalia Krizhanovsky, Paula Czarnowska, Irene Nikkarinen, Andrew Krizhanovsky, Tiago Pimentel, Lucas Torroba Hennigen, Christo Kirov, Garrett Nicolai, Adina Williams, Antonios Anastasopoulos, Hilaria Cruz, Eleanor Chodroff, Ryan Cotterell, Miikka Silfverberg, and Mans Hulden. 2020. SIGMORPHON 2020 shared task 0: Typologically diverse morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 1–39, Online. Association for Computational Linguistics.

Qingyun Wang, Xiaoman Pan, Lifu Huang, Boliang Zhang, Zhiying Jiang, Heng Ji, and Kevin Knight. 2018. Describing a knowledge base. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 10–21, Tilburg University, The Netherlands. Association for Computational Linguistics.

Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. 2020. Towards faithful neural table-to-text generation with content-matching constraints. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1072–1086, Online. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Xinnuo Xu, Ondřej Dušek, Verena Rieser, and Ioannis Konstas. 2021. Agggen: Ordering and aggregating while generating.

Goal-Oriented Script Construction

Qing Lyu* **Li Zhang*** **Chris Callison-Burch**
University of Pennsylvania
{lyuqing,zharry,ccb}@seas.upenn.edu

Abstract

The knowledge of scripts, common chains of events in stereotypical scenarios, is a valuable asset for task-oriented natural language understanding systems. We propose the Goal-Oriented Script Construction task, where a model produces a sequence of steps to accomplish a given goal. We pilot our task on the first multilingual script learning dataset supporting 18 languages collected from wikiHow, a website containing half a million how-to articles. For baselines, we consider both a generation-based approach using a language model and a retrieval-based approach by first retrieving the relevant steps from a large candidate pool and then ordering them. We show that our task is practical, feasible but challenging for state-of-the-art Transformer models, and that our methods can be readily deployed for various other datasets and domains with decent zero-shot performance[1].

1 Introduction

A *script* is a standardized sequence of events about stereotypical activities (Feigenbaum et al., 1981). For example, "*go to a restaurant*" typically involves "*order food*", "*eat*", "*pay the bill*", etc. Such script knowledge has long been proposed as a way to enhance AI systems (Abelson and Schank, 1977). Specifically, task-oriented dialog agents may greatly benefit from the understanding of *goal-oriented scripts*[2]. However, the evaluation of script knowledge remains an open question (Chambers, 2017). Moreover, it is unclear whether current models can generate complete scripts. Such an ability is in high demand for recent efforts to reason about complex events (Li et al., 2020; Wen et al., 2021)[3].

Input: (How to) Obtain Travel Documents

Output:

1. Authority grants approval for travel documents.
2. Applicant takes a photograph if that is required to receive travel document.
3. Travel documents approved and granted to traveler.
4. Get and/or fill out all documents that must be submitted to apply for the travel document.
5. Traveler pays fees and submits application documents.
6. Applicant receives letter, passport, visa stamp, or whatever travel documents were requested.
7. Item is delivered to the destination.
8. Prepare required application documents.
9. Pay any required application fees.

Figure 1: An example script constructed by our Step-Inference-Ordering pipeline in a zero-shot manner. The input is a *goal*, and the output is an ordered list of steps.

We propose the task of *Goal-Oriented Script Construction* (GOSC) to *holistically* evaluate a model's understanding of scripts. Given a *goal* (or the name of a script), we ask the model to construct the sequence of *steps* (or events in a script) to achieve the goal. This task targets a model's ability to narrate an entire script, subsuming most existing evaluation tasks. Our rationale is that a model that *understands* some scripts (e.g. how to "*travel abroad*" and "*go to college*") should be able to produce *new* ones (e.g. how to "*study abroad*") using the absorbed knowledge, close to how humans learn.

While almost all prior script learning work has focused on English, we introduce a novel multilingual corpus. Our corpus is collected from wikiHow (wikihow.com), a website of how-to articles in 18 languages. The articles span a wide range of domains, from commonplace activities like going to a restaurant to more specific ones like protecting

* Equal contribution.

[1] Our models and data are be available at https://github.com/veronica320/wikihow-GOSC.

[2] https://developer.amazon.com/alexaprize

[3] www.darpa.mil/program/knowledge-directed-artificial-intelligence-reasoning-over-schemas

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 184–200,
Aberdeen, Scotland, UK, 20-24 September 2021.

oneself from the coronavirus.

We train and evaluate several baseline systems on our GOSC task. First, we consider a generation-based approach where a pretrained language model, multilingual T5, is finetuned to produce scripts from scratch. As an alternative, observing that most desired steps can be drawn from the training scripts due to their magnitude and high coverage, we also propose a retrieval-based approach. Concretely, we develop a Step-Inference-Ordering pipeline using existing models to retrieve relevant steps and order them. We also improve the pipeline with techniques such as multitask learning. From the experiments, the GOSC task proves challenging but feasible for state-of-the-art Transformers. Furthermore, we show that our pipeline trained on wikiHow can generalize to other datasets and domains (see an example in Figure 1). On three classic script corpora, OMICS, SMILE, and DeScript, it achieves strong zero-shot performance. It can also be directly deployed to construct scripts in distant domains (e.g. military/political).

In this paper, we make several contributions:
1) We propose the GOSC task targeting the comprehensive understanding of scripts.
2) We introduce the first multilingual script learning dataset available in 18 languages.
3) We compare generation-based and retrieval-based approaches using both automatic and human judgments, which demonstrate the feasibility but also the difficulty of GOSC.
4) We show that our approach can be readily applied to other datasets or other domains.

2 Related Work

The notion of *scripts* (Abelson and Schank, 1977), or *schemas* (Rumelhart, 1975), encodes the knowledge of standardized event sequences. We dissect previous work on script learning into two lines, *narrative* and *procedural*.

One line of work focuses on *narrative* scripts, where *declarative*, or *descriptive* knowledge is distilled from narrative texts like news or stories (Mujtaba and Mahapatra, 2019). Such scripts are not goal-oriented, but descriptions of sequential events (e.g. a traffic accident involves a collision, injuries, police intervention, etc.). Chambers and Jurafsky (2008) introduced the classic Narrative Cloze Test, where a model is asked to fill in the blank given a script with one missing event. Following the task, a few papers made extensions on representation (Chambers and Jurafsky, 2009; Pichotta and Mooney, 2014) or modeling (Jans et al., 2012; Pichotta and Mooney, 2016a,c,b), achieving better performance on Narrative Cloze. Meanwhile, other work re-formalized Narrative Cloze as language modeling (LM) (Rudinger et al., 2015) or multiple-choice (Granroth-Wilding and Clark, 2016) tasks. However, the evolving evaluation datasets contain more spurious scripts, with many uninformative events such as "say" or "be", and the LMs tend to capture such cues (Chambers, 2017).

The other line of work focuses on *procedural* scripts, where events happen in a scenario, usually in order to achieve a goal. For example, to "visit a doctor", one should "make an appointment", "go to the hospital", etc. To obtain data, Event Sequence Descriptions (ESD) are collected usually by crowdsourcing, and are cleaned to produce scripts. Thus, most such datasets are small-scale, including OMICS (Singh et al., 2002), SMILE (Regneri et al., 2010), the Li et al. (2012) corpus, and DeScript (Wanzare et al., 2016). The evaluation tasks are diverse, ranging from event clustering, event ordering (Regneri et al., 2010), text-script alignment (Ostermann et al., 2017) and next event prediction (Nguyen et al., 2017). There are also efforts on domain extensions (Yagcioglu et al., 2018; Berant et al., 2014) and modeling improvements (Frermann et al., 2014; Modi and Titov, 2014).

In both lines, it still remains an open problem what kind of automatic task most accurately evaluates a system's understanding of scripts. Most prior work has designed tasks focusing on various fragmented pieces of such understanding. For example, Narrative Cloze assesses a model's knowledge for completing a close-to-finished script. The ESD line of work, on the other hand, evaluates script learning systems with the aforementioned variety of tasks, each touching upon a specific piece of script knowledge nonetheless. Recent work has also brought forth generation-based tasks, but mostly within an open-ended/specialized domain like story or recipe generation (Fan et al., 2018; Xu et al., 2020).

Regarding data source, wikiHow has been used in multiple NLP efforts, including knowledge base construction (Jung et al., 2010; Chu et al., 2017), household activity prediction (Nguyen et al., 2017), summarization (Koupaee and Wang, 2018; Ladhak et al., 2020), event relation classification (Park and Motahari Nezhad, 2018), and next passage completion (Zellers et al., 2019). A few recent papers

(Zhou et al., 2019; Zhang et al., 2020b) explored a set of separate goal-step inference tasks, mostly in binary-classification/multiple-choice formats, with few negative candidates. Our task is more holistic and realistic, simulating an open-ended scenario with retrieval/generation settings. We combine two of our existing modules from Zhang et al. (2020b) into a baseline, but a successful GOSC system can certainly include other functionalities (e.g. paraphrase detection). Also similar is Zhang et al. (2020a), which doesn't include an extrinsic evaluation on other datasets/domains though.

In summary, our work has the following important differences with previous papers:
1) Existing tasks mostly evaluate fragmented pieces of script knowledge, while GOSC is higher-level, targeting the ability to invent *new, complete* scripts.
2) We are the first to study *multilingual* script learning. We evaluate several baselines and make improvements with techniques like multitask learning.
3) Our dataset improves upon the previous ones in multiple ways, with higher quality than the mined narrative scripts, lower cost and larger scale than the crowdsourced ESDs.
4) The knowledge learned from our dataset allows models to construct scripts in other datasets/domains without training.

3 Goal Oriented Script Construction

We propose the Goal-Oriented Script Construction (GOSC) task. Given a *goal* g, a system constructs a complete script as an ordered list of *steps* S, with a ground-truth reference T. As a hint of the desired level of granularity, we also provide an expected number of steps (or length of the script), l, as input. Depending on whether the set of possible candidate steps are given in advance, GOSC can happen in two settings: Generation or Retrieval.

In the **Generation setting**, the model must generate the entire script from scratch.

In the **Retrieval setting**, a large set of candidate steps C is given. The model must predict a subset of steps S from C, and provide their ordering.

4 Multilingual WikiHow Corpus

Our previously wikiHow corpus (Zhang et al., 2020b) is a collection of how-to articles in English (en). We extend this corpus by crawling wikiHow in 17 other languages, including Spanish (es), Portuguese (pt), Chinese (zh), German (de), French (fr), Russian (ru), Italian (it), Indonesian (id), Dutch (nl), Arabic (ar), Vietnamese (vn), Thai (th), Japanese (jp), Korean (ko), Czech (cz), Hindi (hi), and Turkish (tr). The resulting multilingual wikiHow corpus may be used in various tasks in NLP and other fields.

```
{
 "title": "Eat at a Sit Down Restaurant",
 "category": "FOOD AND ENTERTAINING",
 "ordered": True,
 "sections": [ ...
  {
   "section": "Ordering Out",
   "steps": [ ...
    "Order drinks first.",
    "Ask about daily specials.",
    "Look over the menu and place your
     food order.", ...
   ],
  }, ... ]}
```

Figure 2: An abridged example script extracted from the English wikiHow article "How to Eat at a Sit Down Restaurant".

For script learning, we extract from each wikiHow article the following critical components to form a *goal-oriented script*.
Goal: the title stripped of "How to";
Section: the header of a "method" or a "part" which contains multiple steps;[4]
Steps: the headlines of step paragraphs;
Category: the top-level wikiHow category.
An example wikiHow script is shown in Figure 2.

Our previous corpus provides labels of whether each English article is ordered, predicted by a high-precision classifier. We project these labels to other languages using the cross-language links in each wikiHow article. For articles without a match to English, it defaults to unordered. In our task setup, we only require the model to order the steps if an article is ordered.

For all experiments below, we randomly hold out 10% articles in each language as the test set, and use the remaining 90% for training and development.[5]

We use the corpus to construct a dataset for multilingual GOSC. For the Retrieval setting, the set of candidate steps C are all the steps present in the test set. However, we observe that not only the large number of steps may render the evaluation intractable, but most steps are also evidently distant from the given goal. To conserve computing power, we restrict C as all the steps from articles within

[4]We ignore this hierarchical relation and flatten all steps in all Sections as the Steps of the script.

[5]See Appendix A for our corpus statistics.

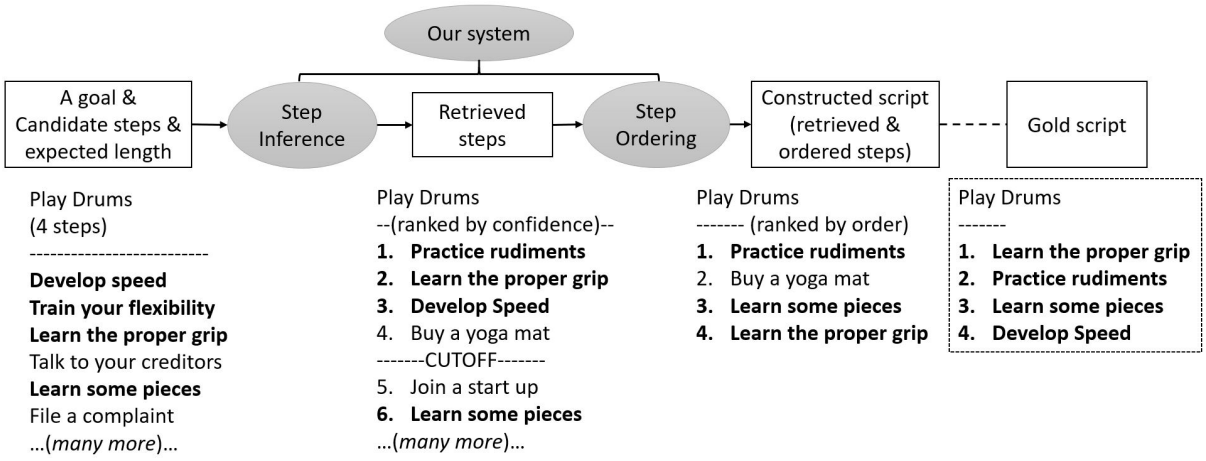


Figure 3: Our Step-Inference-Ordering pipeline for the GOSC Retrieval task. An example *ordered* script is shown with example steps in the input and output. Those that appear in the ground-truth script is in bold.

the same wikiHow category for each script.

5 Models

We develop two systems based on state-of-the-art Transformers for the GOSC task.[6]

5.1 Generation Approach: Multilingual T5

For the Generation setting, we finetune mT5 (Xue et al., 2021), a pretrained generation model that is not only state-of-the-art on many tasks but also the only available massively multilingual one to date.

During finetuning, we provide the goal of each article in the training set as a prompt, and train the model to generate the sequence of all the steps conditioned on the goal. Therefore, the model's behavior is similar to completing the task of inferring relevant steps and sorting them at once. At inference time, the model generates a list of steps given a goal in the test set.

5.2 Retrieval Approach: Step-Inference-Ordering Pipeline

We then implement a *Step-Inference-Ordering pipeline* for the Retrieval setting. Our pipeline contains a Step Inference model to first gather the set of desired steps, and a Step Ordering model to order the steps in the set. These models are based on our previous work (Zhang et al., 2020b). Under the hood, the models are pretrained XLM-RoBERTa (Conneau et al., 2020) or mBERT (Devlin et al., 2019) for binary classification, both state-of-the-art multilingual representations.

Our Step Inference model takes a goal and a candidate step as input, and outputs whether the candidate is indeed a step toward the goal with a confidence score. During training, for every script, its goal forms a positive example along with each of its steps. We then randomly sample 50 steps from other scripts within the same wikiHow category and pair them with the goal as negative examples. The model predicts a label for each goal-step pair with a cross-entropy loss. During evaluation, for each script in the test set, every candidate step is paired with the given goal as the model input. We then rank all candidate steps based on the model confidence scores decreasingly. Finally, the top l steps are retained, where l is the required length.

Our Step Ordering model takes a goal and two steps as input, and outputs which step happens first. During training, we sample every pair of steps in each ordered script as input to the model with a cross-entropy loss. During evaluation, we give every pair of retrieved steps as input, and count the total number of times that a step is ranked before others. We then sort all steps by this count to approximate their complete ordering.

An illustration of our Step-Inference-Ordering pipeline is shown in Figure 3. We also consider two additional variations.

Multitask Learning (MTL): The Step Inference and the Step Ordering models share the encoder layer, but have separate classifier layers. During training, the MTL system is then presented with a batch of examples from each task in an alternating fashion. During evaluation, the corresponding classifier is used.

Cross-Lingual Zero-Shot Transfer (C0): While there are abundant English training scripts, data in some other languages are scarce. Hence, we

[6]Reproducibility details can be found in Appendix C.

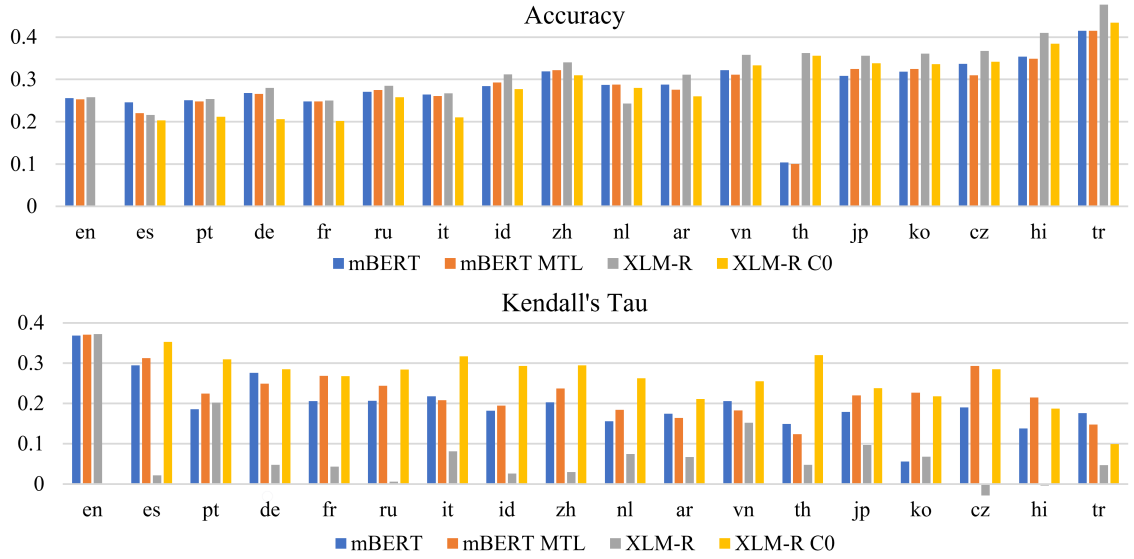


Figure 4: Detailed performance on each language from Table 2.

Lang.	en	es	pt	de	fr	ru
Perp.	17	11	24	97	46	79
Bert.	.823	.702	.682	.677	.718	.682
Lang.	it	id	zh	nl	ar	vn
Perp.	116	269	13,249	955	746	97
Bert.	.653	.692	.667	.690	.701	.695
Lang.	th	jp	ko	cz	hi	tr
Perp.	29,538	73,952	2,357	1,823	2,033	36,848
Bert.	.701	.679	.692	.682	.704	.665

Table 1: Auto evaluation results for the Generation setting (Perplexity and BERTScore F1 measure). The performance of multilingual T5 is reported.

also attempt to directly evaluate the English-trained models on non-English data.

6 In-Domain Evaluation

To demonstrate the performance of models on the GOSC task, we evaluate them on our multilingual wikiHow dataset using both automatic metrics and human judgments. The ultimate utility for this task is the extent to which a human can follow the constructed steps to accomplish the given goal. As direct user studies might be costly and hard to standardize, we carefully choose measures that adhere to this utility. By default, all models are trained and evaluated on the same language.

[7]Multitask XLM-R and cross-lingual zero-shot mBERT are found to perform a lot worse and thus omitted.

Model	English only		Avg. all lang.s	
	Acc.	Kendall's τ	Acc.	Kendall's τ
mBERT	.256	.369	.286	.198
mBERT MTL	.253	.371	.283	.226
XLM-R	.258	.372	.317	.075
XLM-R C0	-	-	.291	.264

Table 2: Auto evaluation results for the Retrieval setting (Accuracy and Kendall's Tau). The performance of mBERT and XLM-RoBERTa, along with their multitask (MTL) and crosslingual zero-shot transfer (C0) variations, are reported [7].

6.1 Auto Evaluation for Generation Setting

To automatically evaluate models in the Generation Setting, we report **perplexity** and **BERTScore** (Zhang et al., 2019), as two frequently used metrics for evaluating text generation.

The mean perplexity of mT5 on the test set of each language is shown in Table 1. The results show a large range of variation. To see if perplexity correlates with the data size, we conduct a Spearman's rank correlation two-tailed test. We find a Spearman's ρ of -0.856 and a p-value of $1e-5$ between the perplexity and the number of articles in each language in our dataset; we find a Spearman's ρ of -0.669 and a p-value of $2e-4$ between the perplexity and the number of tokens in each language in the mC4 corpus where mT5 is pretrained on. These statistics suggest a significant correlation between perplexity and data size, while other typological factors are open to investigation.

Table 1 also shows the BERTScore F1 measure

of the generated scripts compared against the gold scripts. Except for English (.82), the performance across different languages varies within a relatively small margin (.65 - .72). However, we notice that as a metric based on the token-level pairwise similarity, BERTScore may not be the most suitable metric to evaluate scripts. It is best designed for *aligned* texts (e.g. a machine-translated sentence and a human-translated one), whereas in scripts, certain candidate steps might not have aligned reference steps. Moreover, BERTScore does not measure whether the ordering among steps is correct. To address these flaws, we further perform human evaluation in Section 6.3.

6.2 Auto Evaluation for Retrieval Setting

To automatically evaluate models in the Retrieval Setting, we first calculate **accuracy**, i.e. the percentage of predicted steps that exist in the ground-truth steps. To account for the ordering of steps, we also compute **Kendall's** τ between the overlapping steps in the prediction and the ground-truth.

The performance of our Step Inference-Ordering pipeline using mBERT and XLM-RoBERTa[8] on all 18 languages are shown in Figure 4. Complete results can be found in Appendix D. Across languages, the results are generally similar with a large room for improvement. On average, our best system constructs scripts with around 30% accuracy and around 0.2 Kendall's τ compared to the ground-truth. Compared to the baseline, our multi-task and cross-lingual zero-shot variations demonstrate significant improvement on ordering. This is especially notable in low-resource languages. For example, MTL on Korean and C0 on Thai both outperform their baseline by 0.17 on Kendall's τ.

6.3 Human Evaluation

To complement automatic evaluation, we ask 6 annotators[9] to each *edit* 30 output scripts by the Step-Inference-Ordering pipeline and mT5 in English, French, Chinese, Japanese, Korean and Hindi, respectively. The *edit* process consists of a sequence of two possible actions: either 1) delete a generated step entirely if it is irrelevant, nonsensical or not a reasonable step of the given goal, or 2) move a step somewhere else, if the order is incorrect. Then, the generated script is evaluated against the edited script in 3 aspects:

Correctness, approximated by the length (number of steps) of the edited script over that of the originally constructed script (c.f. precision);

Completeness, approximated by the length of the edited script over that of the ground-truth script (c.f. recall);

Orderliness, approximated by Kendall's τ between overlapping steps in the edited script and the generated script.[10]

Retrieval: Step-Inference-Ordering pipeline						
Language	en	fr	zh	jp	ko	hi
Correctness	.70	.39	.50	.49	.45	.82
Completeness	.70	.39	.50	.49	.45	.82
Orderliness	.45	.38	.16	.12	.10	.75
Generation: mT5						
Language	en	fr	zh	jp	ko	hi
Correctness	.39	.51	.46	.40	.37	.49
Completeness	.35	.40	.46	.30	.36	.41
Orderliness	.82	.46	.60	.81	.69	.88

Table 3: Human judgments of correctness, completeness and orderliness of the output of the Step-Inference-Order pipeline and the mT5 model for the same set of 30 gold scripts, in six languages.

The results are shown in Table 3. While the constructed scripts in the Retrieval setting contain more correct steps, their ordering is significantly worse than those in the Generation setting. This suggests that the generation model is better at producing *fluent* texts, but can easily suffer from hallucination.

6.4 Qualitative Examples

To understand models' behavior, we present two representative scripts produced by the mBERT Retrieval model and the mT5 Generation model side by side, accompanied by the ground-truth script, shown in Figure 5.

The retrieved "Draw Santa Claus" script has a high step accuracy (85%), with a reasonable ordering of drawing first the outline and then details. The generation output is more off-track, hallucinating irrelevant details like "singing" and "scorpion", despite being on the general topic of drawing. It also generates more repetitive steps (e.g. the head is drawn twice), most of which are abridged.

As for "Make a Quotebook", the retrieved script

[8] XLM-RoBERTa is not able to converge on the training data for Step Ordering for all but 3 languages using a large set of hyperparameter combinations.

[9] The annotators are graduate students and native or proficient speakers of the language assigned.

[10] In this formulation, the correctness and completeness of a retrieval-based model are equal, since the length of its constructed script is equal to that of the ground truth script by definition.

Script Name: Draw Santa Claus

Retrieval Output

1. Sketch in Santa's clothes.
2. Go back over and outline Santa's body.
3. Outline Santa's head.
4. Draw Santa's legs.
5. Draw Santa's beard.
6. Draw Santa's hat.
7. Color in your Santa.
8. Draw Santa's eyes.
9. Add Santa.
10. Give Santa some eyebrows.
11. Begin with the nose.
12. Make the nose.
13. Draw the mustache.
14. Add the arms and hands.

Generation Output (abridged)

1. Practice singing with children and ask them to draw in bells and an ear.
2. Draw the hands using slightly curved lines.
3. On the face, sketch a medium circle for Santa's head.
4. On the body for the legs, draw a slightly slanted oval for the body.
5. Draw a similar length triangle for the face, above the lines and to the right side of the base of the body.
6. Draw smaller ovals for hands in depicting a trapezoid for a larger scorpion.
7. Trace with a pen and erase unnecessary lines.
8. Color to your liking!
9. Draw another Santa Claus head.
10. ...(repeated)

Gold Reference

1. Outline Santa's head.
2. Draw two large circles for the body.
3. Add the arms and hands.
4. Draw Santa's legs.
5. Begin with the nose.
6. Draw the mustache.
7. Draw Santa's eyes.
8. Give Santa some eyebrows.
9. Draw Santa's beard.
10. Draw Santa's hat.
11. Draw the mouth.
12. Go back over and outline Santa's body.
13. Sketch in Santa's clothes.
14. Color in your Santa.

Script Name: Make a Quote Book

Retrieval Output

1. Find a blank book to record your quotes in.
2. Decide how you want to write out quotes.
3. Look up quotations from popular shows, movies, and songs.
4. Unpack unusual or poetic phrases one at a time.

Generation Output (abridged)

1. Insert a quote when you need to include a word or clause in it. ...
2. Add a right emphasis to highlight important words in a quote. ...
3. Show your quote to others. ...
4. Make sure you spell your quote correctly. ...
5. Cite sourced texts in nonfiction. ...

Gold Reference

1. Find a blank book to record your quotes in.
2. Find some quotes.
3. Decide how you want to write out quotes.
4. Decorate it!

Figure 5: Two example scripts constructed by our Retrieval and Generation approaches.

has a 50% step accuracy. The third step, though not in the gold reference, is similar enough to "find some quotes", suggesting that our exact match evaluation isn't perfect. In the generated script, all steps are also generally plausible, but some essential steps are missing (e.g. find a book, find quotes). This suggests that the generation model dwells too much on the details, ignoring the big picture.

These patterns in the two scripts are common in the model outputs, a larger sample of which is included in the Supplementary Materials.

7 Zero-shot Transfer Learning

To show the potential of our model for transfer learning, we use the retrieval-based Step-Inference-Ordering pipeline finetuned on wikiHow to construct scripts for other datasets and domains. We quantitatively evaluate our model on 4 other script learning corpora, and qualitatively analyze some constructed scripts in a case study.

7.1 Quantitative Evaluation

Since no multilingual script data are available yet, we perform transfer learning experiments on 4 other English script corpora, OMICS (Singh et al., 2002), SMILE (Regneri et al., 2010), DeScript (Wanzare et al., 2016) [11], and the KAIROS Schema Learning Corpus (LDC2020E25). The first 3 pertain to human activities, while the last is in the military and political domain. They are all in the format of different *scenarios* (e.g. "eat in a restaurant", similar to our *goal*) each with a number of *event sequence descriptions* (ESDs, similar to our *steps*). Statistics for each corpus are in Table 4.

Corpus	Corpus Stats.		Results	
	Scenarios	ESDs	Acc.	Kendall's τ
SMILE	22	386	.435	.391
OMICS	175	9044	.346	.443
DeScript	40	4000	.414	.418
KAIROS	28	28	.589	.381

Table 4: The zero-shot GOSC Retrieval performance of XLM-RoBERTa finetuned on wikiHow on 4 target corpora.

For each dataset, we select the ESD with the most steps for every scenario as a representative script to avoid duplication, thus converting the dataset to a GOSC evaluation set under the Retrieval setting. We then use the XLM-RoBERTa-based Step-Inference-Ordering pipeline trained on our English wikiHow dataset to directly construct scripts on each target set, and report its zero-shot performance in Table 4. We see that $30\% - 60\%$ steps are accurately retrieved, and around 40% are correctly ordered. This is close to or even better than the in-domain results on our English test set. As a comparison, a random baseline would have only 0.013 Accuracy and 0.004 τ on average. Both facts indicate that the script knowledge learned from our dataset is clearly non-trivial.

[11] The above 3 corpora are all obtained from http://www.coli.uni-saarland.de/projects/smile/

7.2 Case Study: The *Bombing Attack* Scripts

To explore if the knowledge about *procedural* scripts learned from our data can also facilitate the zero-shot learning of *narrative* scripts, we present a case study in the context of the DARPA KAIROS program[12]. One objective of KAIROS is to automatically induce scripts from large-scale narrative texts, especially in the military and political domain. We show that models trained on our data of commonplace events can effectively transfer to vastly different domains.

With the retrieval-based script construction model finetuned on wikiHow, we construct five scripts with different granularity levels under the *Improvised Explosive Device (IED) attack* scenario: "Roadside IED attack", "Backpack IED attack", "Drone-brone IED attack", "Car bombing IED attack", "IED attack". We take the name of each script as the input *goal*, and a collection of related documents retrieved from Wikipedia and Voice of America news as data sources for extracting step candidates.

Our script construction approach has two components. First, we extract all events according to the KAIROS Event Ontology from the documents using OneIE (Lin et al., 2020). The ontology defines 68 event primitives, each represented by an *event type* and multiple *argument types*, e.g. a Damage-type event has arguments including Damager, Artifact, Place, etc. OneIE extracts all event instances of the predefined primitives from our source documents. Each event instance contains a *trigger* and several *arguments* (e.g. Trigger: "destroy", Damager: "a bomber", Artifact: "the building", ...). All event instances form the candidate pool of steps for our target script.

Since the events are represented as trigger-arguments tuples, a conversion to the raw textual form is needed before inputting them into our model. This is done by automatically instantiating the corresponding event type template in the ontology with the extracted arguments. If an argument is present in the extracted instance, we directly fill it in the template; else, we fill in a placeholder word (e.g."some", "someone", depending on the argument type). For example, the template of Damage-type events is "$\langle arg1 \rangle$ damaged $\langle arg2 \rangle$ using $\langle arg3 \rangle$ instrument", which can be instantiated as "A bomber damaged the building using some instrument"). Next, we run the Step Inference-Ordering Pipeline in Section 5.2 on the candidate pool given the "goal". The only modification is that since we don't have a gold reference script length in this case, all retrieved steps with a confidence score higher than a threshold (default=0.95) are retained in the final script.

Script Name: Roadside Improvised Explosive Device Attack

1. Movement.Transportation
   (Example: "Taliban transported explosives in car to centre place.")
2. Conflict.Attack.DetonateExplode
   (Example: "Someone detonated ied explosive device at ghazni place.")
3. Conflict.Attack
   (Example: "Group attacked convoy using explosive.")
4. Life.Injure
   (Example: "861 was injured by contractor using explosive.")
5. ArtifactExistence.DamageDestroyDisableDismantle.Damage
   (Example: "Someone damaged vehicle.")
6. ArtifactExistence.ManufactureAssemble
   (Example: "Someone manufactured or assembled or produced weapons.")
7. Life.Die
   (Example: "Members died, killed by efps killer.")
8. Justice.TrialHearing
   (Example: "Someone tried someone before dunford court or judge.")

Figure 6: An example narrative script produced by our retrieval-based pipeline trained on wikiHow. Each event is represented by its Event Type and an example sentence.

We manually evaluate the constructed scripts with the metrics defined in Section 6.3, except *Completeness* as we don't have gold references. The 5 constructed scripts have an average *Correctness* of 0.735 and *Orderliness* of 0.404. Despite the drastic domain shift from wikiHow to KAIROS, our model can still exploit its script knowledge to construct scripts decently. An example script, "Roadside IED attack", is shown in Figure 6. All the steps retrieved are sensible, and most are ordered with a few exceptions (e.g. the ManufactureAssemble event should precede all others).[13]

8 Limitations

Event representation: Our representation of goals and steps as natural language sentences, though containing richer information, brings the extra difficulty in handling steps with similar meanings. For example, "change strings frequently" and "put on new strings regularly" have nearly identical meanings and both are correct steps for the goal "maintain a guitar". Hence, both could be included by a retrieval-based model, which is not desired.

[12] www.darpa.mil/program/knowledge-directed-artificial-intelligence-reasoning-over-schemas

[13] More details on the format of the script, all five constructed scripts, the event ontology, and a list of news documents used can be found in the Supplementary Materials.

Modeling: Since GOSC is a new task, there is no previously established SOTA to compare with. We build a strong baseline for each setting, but they are clearly not the necessary or sufficient means to do the task. For example, our Step-Inference-Ordering pipeline would benefit from a paraphrasing module that eliminates semantic duplicates in retrieved steps. It also currently suffers from long run-time especially with a large pool of candidates, since it requires pairwise goal-step inference. An alternative is to filter out most irrelevant steps using similarity-based heuristics in advance.

Evaluation: Under the retrieval-based setting, our automatic evaluation metrics do not give credit to inexact matches as discussed above, which can also be addressed by a paraphrasing module. Meanwhile, for the generation-based setting, BERTScore, or other comparison-based metrics like BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014), may not be the most suitable metric to evaluate scripts. They are best designed for *aligned* texts like translation pairs, and do not measure whether the ordering among steps is correct. While we complement it with manual evaluation, only one human annotator is recruited for each language, resulting in potential subjectivity. Alternatively, crowdsourcing-based evaluation is costly and hard to standardize. Due to the complexity of the GOSC task and its evaluation, we suggest that future work investigate better means of evaluation.

9 Conclusion and Future Work

We propose the first multilingual script learning dataset and the first task to evaluate the holistic understanding of scripts. By comprehensively evaluating model performances automatically and manually, we show that state-of-the-art models can produce complete scripts both in- and out-of-domain, with a large room for improvement. Future work should investigate additional aspects of scripts, such as usefulness, granularity, etc., as well as their utility for downstream tasks that require automated reasoning.

Acknowledgments

This research is based upon work supported in part by the DARPA KAIROS Program (contract FA8750-19-2-1004), the DARPA LwLL Program (contract FA8750-19-2-0201), and the IARPA BETTER Program (contract 2019-19051600004). Approved for Public Release, Distribution Unlimited. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, IARPA, or the U.S. Government.

Special thanks go to our annotators: Artemis Panagopoulou, Daniel Joongwon Kim, Simmi Mourya, and Liam Dugan. We also thank Daphne Ippolito for the help on mT5; Ying Lin, Sha Li, Zhenhailong Wang, Pengfei Yu, Tuan Lai, Haoyang Wen, and Heng Ji for providing the source documents and the OneIE results for our case study. Finally, we appreciate the valuable feedback from the anonymous reviewers.

References

Robert Abelson and Roger C Schank. 1977. Scripts, plans, goals and understanding. *An inquiry into human knowledge structures New Jersey*, 10.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510.

Nathanael Chambers. 2017. Behind the scenes of an evolving event cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 41–45.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610.

Cuong Xuan Chu, Niket Tandon, and Gerhard Weikum. 2017. Distilling task knowledge from how-to communities. In *Proceedings of the 26th International Conference on World Wide Web*, pages 805–814.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

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

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Edward A Feigenbaum, Avron Barr, and Paul R Cohen. 1981. *The handbook of artificial intelligence*. Addison-Wesley.

Lea Frermann, Ivan Titov, and Manfred Pinkal. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 49–57.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 336–344.

Yuchul Jung, Jihee Ryu, Kyung-min Kim, and Sung-Hyon Myaeng. 2010. Automatic construction of a large-scale situation ontology by mining how-to instructions from the web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(2-3):110–124.

Mahnaz Koupaee and William Yang Wang. 2018. WikiHow: A large scale text summarization dataset. *ArXiv*, abs/1810.09305.

Faisal Ladhak, Esin Durmus, Claire Cardie, and Kathleen McKeown. 2020. WikiLingua: A new benchmark dataset for cross-lingual abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4034–4048, Online. Association for Computational Linguistics.

Boyang Li, Stephen Lee-Urban, Darren Scott Appling, and Mark O Riedl. 2012. Crowdsourcing narrative intelligence. *Advances in Cognitive systems*, 2(1).

Manling Li, Qi Zeng, Ying Lin, Kyunghyun Cho, Heng Ji, Jonathan May, Nathanael Chambers, and Clare Voss. 2020. Connecting the dots: Event graph schema induction with path language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 684–695.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 49–57.

Dena Mujtaba and Nihar Mahapatra. 2019. Recent trends in natural language understanding for procedural knowledge. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 420–424. IEEE.

Dai Quoc Nguyen, Dat Quoc Nguyen, Cuong Xuan Chu, Stefan Thater, and Manfred Pinkal. 2017. Sequence to sequence learning for event prediction. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 37–42, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Simon Ostermann, Michael Roth, Stefan Thater, and Manfred Pinkal. 2017. Aligning script events with narrative texts. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 128–134.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Hogun Park and Hamid Reza Motahari Nezhad. 2018. Learning procedures from text: Codifying how-to procedures in deep neural networks. In *Companion Proceedings of the The Web Conference 2018*, pages 351–358.

Karl Pichotta and Raymond Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–229.

Karl Pichotta and Raymond Mooney. 2016a. Statistical script learning with recurrent neural networks. In *Proceedings of the Workshop on Uphill Battles in*

Language Processing: Scaling Early Achievements to Robust Methods, pages 11–16.

Karl Pichotta and Raymond Mooney. 2016b. Using sentence-level lstm language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–289.

Karl Pichotta and Raymond J Mooney. 2016c. Learning statistical scripts with lstm recurrent neural networks. In *AAAI*, pages 2800–2806.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1686.

David E Rumelhart. 1975. Notes on a schema for stories. In *Representation and understanding*, pages 211–236. Elsevier.

Push Singh, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1223–1237. Springer.

Lilian DA Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2016. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge. In *Proceedings of the tenth international conference on language resources and evaluation (LREC'16)*, pages 3494–3501.

Haoyang Wen, Ying Lin, Tuan Lai, Xiaoman Pan, Sha Li, Xudong Lin, Ben Zhou, Manling Li, Haoyu Wang, Hongming Zhang, Xiaodong Yu, Alexander Dong, Zhenhailong Wang, Yi Fung, Piyush Mishra, Qing Lyu, Dídac Surís, Brian Chen, Susan Windisch Brown, Martha Palmer, Chris Callison-Burch, Carl Vondrick, Jiawei Han, Dan Roth, Shih-Fu Chang, and Heng Ji. 2021. RESIN: A dockerized schema-guided cross-document cross-lingual cross-media information extraction and event tracking system. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 133–143, Online. Association for Computational Linguistics.

Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. 2020. MEGATRON-CNTRL: Controllable story generation with external knowledge using large-scale language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2831–2845, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498.

Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. 2018. Recipeqa: A challenge dataset for multimodal comprehension of cooking recipes. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1358–1368.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Hongming Zhang, Muhao Chen, Haoyu Wang, Yangqiu Song, and Dan Roth. 2020a. Analogous process structure induction for sub-event sequence prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1541–1550, Online. Association for Computational Linguistics.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020b. Reasoning about goals, steps, and temporal ordering with WikiHow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639, Online. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yilun Zhou, Julie Shah, and Steven Schockaert. 2019. Learning household task knowledge from WikiHow descriptions. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 50–56, Macau, China. Association for Computational Linguistics.

A Corpus Statistics

Table 5 shows the statistics of our multilingual wikiHow script corpus.

Language	en	es	pt	de	fr	ru	it	id	zh
Num. articles	112,111	64,725	34,194	31,541	26,309	26,222	22,553	21,014	14,725
Num. ordered articles	54,852	26,620	7,408	10,681	6,834	5,335	4,308	5,536	4,784
Avg. num. of sections / article	2.5	2.6	3.1	2.7	2.9	3.1	3.1	2.9	2.8
Avg. num. of steps / article	13.7	14.4	16.1	15.1	15.7	16.4	16.2	16.0	15.4
Num. of articles for train/dev	100,900	58,253	30,775	28,387	23,679	23,600	20,298	18,913	13,253
Num. of articles for test	11,211	6,472	3,419	3,154	2,630	2,622	2,255	2,101	1,472

Language	nl	ar	vn	th	jp	ko	cz	hi	tr
Num. articles	13,343	12,157	6,949	5,821	5,567	5,179	5,043	3,104	1,434
Num. ordered articles	2,113	2,567	1,157	1,244	1,209	917	920	888	520
Avg. num. of sections / article	3.1	3.0	3.2	3.1	3.1	3.2	3.1	3.0	3.2
Avg. num. of steps / article	16.2	16.4	17.1	17.7	16.8	17.5	16.4	16.8	19.2
Num. of articles for train/dev	12,009	10,942	6,255	5,239	5,011	4,662	4,539	2,794	1,291
Num. of articles for test	1,334	1,215	694	582	556	517	504	310	143

Table 5: Statistics of our multilingual wikiHow corpus by language, ordered by the number of articles in each language. Each article is converted to a script, including all steps from all sections.

B Evaluation Details

In Section 3, we formalize the Goal-Oriented Script Construction (GOSC) task as follows: Given a *goal* g, the model is asked to construct a complete script as an ordered list of *steps* S, with a ground-truth reference T. As a hint of the desired level of granularity, we also provide an expected number of steps (or length of the script), l, as input.

In the **Retrieval setting**, a set of candidate steps C is also available. We evaluate an output script from two angles: content and ordering.

First, we calculate the accuracy, namely the percentage of predicted steps that exist in the ground-truth. Denote s_i as the i-th step in S.

$$\text{acc} = (\sum_{i}^{l}[s_i \in T])/l$$

If the gold script is ordered, we further evaluate the ordering of the constructed script by calculating Kendall's τ between the intersection of the predicted steps and the ground-truth steps.

$$\tau = \frac{NC(S \cap T, T \cap S) - ND(S \cap T, T \cap S)}{\binom{l}{2}}$$

where NC is the number of concordant pairs, ND the number of discordant pairs; $A \cap B$ is used as a special notation for the intersection of ordered lists, denoting elements that appear in both A and B, in the order of A.

It is likely that a model includes two modules: a retrieval module and an ordering module. In this case, it is sensible to separately evaluate these two modules.

To evaluate the retrieval module independently, assume that the model retrieves a large set of steps R ranked by their *relevance* to the goal g. Denote r_i as the i-th step in R. We calculate recall and normalized discounted cumulative gain[14] at position k. Assume $k > l$.

$$recall_k = (\sum_{i}^{k}[r_i \in T])/k$$

$$NDCG_k = \frac{\sum_{i=1}^{k} \frac{2^{[r_i \in T]}-1}{\log_2(i+1)}}{\sum_{i=1}^{l} \frac{2^1-1}{\log_2(i+1)} + \sum_{i=l+1}^{k} \frac{2^0-1}{\log_2(i+1)}}$$

To evaluate the ordering module independently, we directly give the model the set of ground-truth steps to predict an ordering. We again use Kendall's τ to evaluate the ordered steps.

$$\tau = \frac{NC(T', T) - ND(T', T)}{\binom{l}{2}}$$

where T' is the set ground-truth steps ordered by the model.

In the **Generation** setting, a model is evaluated using perplexity on the test set, following standard practice.

$$perplexity(S) = \exp\left(-\frac{L(S)}{\text{count of tokens in } S}\right)$$

where $L(S)$ is the log-likelihood of the sequence of steps assigned by the model.

When evaluating a model on multiple scripts, all aforementioned metrics are averaged.

[14] We set the true relevance of each predicted step as 1 if it exists in the ground-truth steps, and 0 otherwise.

C Modeling Details

All our models are implemented using the HuggingFace Transformer service[15]. For all experiments, we hold out 5% of the training data for development.

The pretrained models we use include: the `bert-base-multilingual-uncased` checkpoint (168M parameters) for mBERT, the `xlm-roberta-base` checkpoint (270M parameters) for XLM-RoBERTa, the `roberta-base` checkpoint (125M parameters) for RoBERTa[16], and the `mT5-Large` checkpoint (1B parameters) for mT5[17].

For mBERT, XLM-RoBERTa and RoBERTa, we finetune the pretrained models on our dataset using the standard `SequenceClassification` pipeline on HuggingFace[18]. For mT5, we refer to the official finetuning scripts[19] from the project's Github repository.

For each in-domain evaluation experiment, we perform grid search on learning rate from $1e-5$ to $5e-8$, batch size from 16 to 128 whenever possible, and the number of epochs from 3 to 10. As mBERT and XLM-RoBERTa have a large number of hyperparameters, most of which remain default, we do not list them here. Instead, the hyperparameter values and pretrained models will be available publicly via HuggingFace model sharing. We choose the model with the highest validation performance to be evaluated on the test set. For the Retrieval setting, we consider the accuracy of contracted scripts; for the Generation setting, we consider perplexity.

We run our experiments on an NVIDIA GeForce RTX 2080 Ti GPU, with half-precision floating point format (FP16) with O1 optimization. The experiments in the Retrieval setting take 3 hours to 5 days in the worst case for all languages. The experiments in the Generation setting take 2 hours to 1 day in the worst case for all languages.

[15] https://github.com/huggingface/transformers

[16] The above 3 models are available at https://huggingface.co/transformers/pretrained_models.html

[17] https://github.com/google-research/multilingual-t5

[18] https://huggingface.co/transformers/model_doc/auto.html?highlight=sequence%20classification#transformers.AutoModelForSequenceClassification

[19] https://colab.research.google.com/github/google-research/text-to-text-transfer-transformer/blob/master/notebooks/t5-trivia.ipynb

Lang.	en	es	pt	de	fr	ru
Acc.	.253	.220	.248	.266	.248	**.275**
τ	**.371**	**.313**	**.225**	**.249**	**.269**	.244
Lang.	it	id	zh	nl	ar	vn
Acc.	.261	**.293**	**.322**	**.288**	.276	.311
τ	.208	**.195**	**.237**	**.184**	.164	.183
Lang.	th	jp	ko	cz	hi	tr
Acc.	.100	**.325**	**.325**	.310	.349	.415
τ	.124	**.220**	**.227**	**.293**	**.215**	.148

Table 6: The GOSC Retrieval performance of multitask learning mBERT. Results higher than those produced by the single-task mBERT are in bold.

D Additional Results

Our complete in-domain evaluation results can be found in Table 6, 7, and 8.

E More Qualitative Examples

Aside from the examples shown in Section 6.4, we show 2 more example scripts constructed by the mBERT baseline under the Retrieval setting in Section 5.2 vs. those by the mT5 baseline under the Generation setting in Section 5.1. For each script name, the Retrieval output and the Generation output are shown side by side. Please see Figure 7 and 8 for English examples, and Figure 9 and 10 for Chinese ones.

For more examples, please see the Supplementary Materials. We include 20 examples for each language for the in-domain evaluation, and all 5 examples for the out-of-domain case study on the *Bombing Attack* scenario.

Lang.	Step Retrieval				Ordering	Script Construction	
	Recall@25	Recall@50	NDCG@25	NDCG@50	Kendall's τ	Accuracy	Kendall's τ
en	.337 / .342	.424 / .429	.660 / .660	.648 / .648	.368 / .375	.256 / .258	.369 / .372
es	.319 / .397	.403 / .786	.653 / .532	.642 / .571	.321 / .022	.246 / .216	.295 / .022
pt	.313 / .319	.401 / .412	.679 / .672	.664 / .659	.207 / .212	.251 / .254	.186 / .202
de	.337 / .350	.421 / .438	.687 / .707	.676 / .692	.260 / .026	.268 / .280	.276 / .048
fr	.315 / .320	.405 / .411	.673 / .672	.661 / .659	.244 / .020	.248 / .250	.206 / .043
ru	.336 / .353	.423 / .446	.701 / .715	.688 / .701	.181 / .042	.271 / .285	.207 / .006
it	.332 / .333	.424 / .431	.700 / .705	.686 / .687	.184 / .035	.264 / .267	.218 / .081
id	.351 / .383	.435 / .480	.712 / .744	.699 / .725	.190 / .011	.284 / .312	.182 / .026
zh	.401 / .429	.498 / .536	.750 / .753	.732 / .737	.260 / .027	.319 / .340	.203 / .030
nl	.354 / .382	.447 / .758	.721 / .546	.708 / .597	.179 / .011	.287 / .243	.156 / .075
ar	.351 / .381	.447 / .485	.710 / .735	.694 / .717	.161 / .055	.288 / .311	.175 / .067
vn	.381 / .436	.464 / .544	.769 / .784	.753 / .766	.170 / .171	.322 / .358	.206 / .152
th	.146 / .448	.273 / .566	.330 / .784	.369 / .764	.106 / .056	.104 / .362	.149 / .048
jp	.383 / .447	.487 / .579	.754 / .766	.732 / .751	.170 / .107	.308 / .356	.179 / .097
ko	.381 / .435	.474 / .553	.762 / .780	.744 / .766	.154 / .044	.318 / .361	.056 / .068
cz	.416 / .456	.532 / .582	.772 / .776	.751 / .758	.211 / -.007	.337 / .367	.190 / -.028
hi	.421 / .484	.530 / .610	.782 / .814	.763 / .798	.156 / .029	.354 / .410	.138 / -.004
tr	.509 / .577	.676 / .718	.859 / .881	.829 / .854	.154 / .014	.415 / .477	.176 / .047
Mean	.355 / .404	.454 / .542	.704 / .724	.691 / .714	.204 / .069	.286 / .317	.198 / .075

Table 7: The GOSC Retrieval performance of mBERT and XLM-RoBERTa, divided by a slash in each cell. Both the performance of individual modules and that of script construction are reported.

Lang.	Step Retrieval				Ordering	Script Construction	
	Recall@25	Recall@50	NDCG@25	NDCG@50	Kendall's τ	Accuracy	Kendall's τ
es	0.270	0.338	0.567	0.564	0.360	0.203	0.353
pt	0.265	0.339	0.595	0.590	0.276	0.212	0.310
de	0.271	0.346	0.556	0.558	0.264	0.206	0.285
fr	0.253	0.319	0.585	0.580	0.283	0.202	0.268
ru	0.313	0.390	0.672	0.661	0.252	0.258	0.284
it	0.264	0.338	0.575	0.574	0.268	0.210	0.317
id	0.338	0.424	0.681	0.670	0.321	0.277	0.293
zh	0.379	0.471	0.718	0.706	0.318	0.310	0.295
nl	0.340	0.424	0.684	0.673	0.280	0.280	0.263
ar	0.319	0.400	0.643	0.635	0.235	0.260	0.211
vn	0.392	0.480	0.748	0.733	0.249	0.333	0.255
th	0.418	0.520	0.771	0.753	0.307	0.356	0.320
jp	0.403	0.512	0.751	0.733	0.232	0.338	0.238
ko	0.391	0.485	0.767	0.749	0.182	0.336	0.218
cz	0.406	0.519	0.749	0.732	0.300	0.342	0.285
hi	0.449	0.557	0.770	0.754	0.205	0.384	0.187
tr	0.505	0.646	0.805	0.787	0.167	0.434	0.099
Mean	0.357	0.448	0.692	0.681	0.259	0.296	0.258

Table 8: The GOSC Retrieval performance of XLM-RoBERTa using cross-lingual zero-shot transfer trained on the English data. Both the performance of individual modules and that of script construction are reported.

Script Name: Ice an Injured Ankle

Retrieval Output

1. Place an ice pack on the injured ankle for 15–20 minutes.
2. Rest the injured ankle as much as possible for 48 hours.
3. Compress the injured ankle for 48 hours to prevent swelling.
4. Elevate the injured ankle above your heart to decrease any swelling.
5. Go to the emergency room if the injured ankle cannot bear weight.
6. Apply a compression bandage to a sprained ankle.
7. Consult with a doctor if the ankle does not improve after 2–3 days.
8. Ask a friend to grab your ankles from the bottom.

Generation Output

1. Hang the injured ankle up in a large, open space.
2. Wrap the injured ankle up in a heated towel. Pad your ankle with a bandage.
3. Put the ice pack against your leg for 15-20 minutes.
4. Elevate your leg above the level of irritation.
5. Take an over-the-counter (OTC) pain reliever.
6. See a doctor if 2-3 days persist or worsen.
7. Keep your foot elevated to avoid further swelling.
8. Use an orthopedic shoes to trim your ankle.
9. Wear the right shoes and footwear.

Gold Reference

1. Rest the injured ankle as much as possible for 48 hours.
2. Place an ice pack on the injured ankle for 15–20 minutes.
3. Apply ice 4–8 times daily for the first 48 hours following the injury.
4. Compress the injured ankle for 48 hours to prevent swelling.
5. Elevate the injured ankle above your heart to decrease any swelling.
6. Consult with a doctor if the ankle does not improve after 2–3 days.
7. Take NSAID pain relievers to help with swelling and manage the pain.
8. Go to the emergency room if the injured ankle cannot bear weight.

Figure 7: The “Ice an injured Ankle” script.

Script Name: Solve a Simple Linear Inequality

Retrieval Output

1. Understand the inequality signs.
2. Move the variable to one side of the inequality.
3. Approach the inequality as you would an equation.
4. Combine like terms, or otherwise simplify the inequality.
5. Solve this inequality: 3x+2<−3x+6+5x.
6. Reverse the inequality sign whenever you take the reciprocal of both sides.
7. Solve this inequality in which you need to reverse the sign: −6x−18>12.
8. Solve this compound inequality: 14<2x+4<22.
9. Reverse the inequality sign whenever you multiply or divide by a negative number.
10. Write, then, equation 3 as (see the image above)

Generation Output

1. Recognize that the line is not in the standard form of two numbers plus one.
2. Identify the product of the linear inequality.
3. Find the product of the linear inequality.
4. Do this for all the products of the linear inequality.
5. Substitute in our terms for a product that is in the standard form of two.
6. Choose a more economic alternative to a simple inequality and make them distasteful for your own values.
7. Read the problem and consider solving it for truth.
8. Check your work.
9. Take a break if you made a mistake.

Gold Reference

1. Understand the inequality signs.
2. Combine like terms, or otherwise simplify the inequality.
3. Move the variable to one side of the inequality.
4. Isolate the variable.
5. Approach the inequality as you would an equation.
6. Reverse the inequality sign whenever you multiply or divide by a negative number.
7. Reverse the inequality sign whenever you take the reciprocal of both sides.
8. Solve this inequality: 3x+2<−3x+6+5x.
9. Solve this inequality in which you need to reverse the sign: −6x−18>12.
10. Solve this compound inequality: 14<2x+4<22.

Figure 8: The “Solve a Simple Linear Inequality” script.

Script Name: 竞选美国总统 (Run for the US President)

Retrieval Output

1. 总统必须满足在美国本土出生、至少年满三十五岁、在美国至少已居住十四年以上的条件。
2. 设立总统竞选考察委员会。
3. 让自己出现在候选人列表上。
4. 注册成为正式候选人。
5. 开始竞选。
6. 创建竞选纲领。
7. 建立自己的竞选平台。
8. 你必须提高你的实力。
9. 排序。
10. 考虑组织的愿景。
11. 教英语。
12. 用“领导”代替“管理”。
13. 接受教育。
14. 签订合约。

Generation Output

1. 证明自己是合法出生于美国的公民。
2. 年满35岁。
3. 参加竞选前必须至少在美国连续居住14年。
4. 学习美国总统史。
5. 与家人及支持者仔细商讨。
6. 筹备竞选研究委员会。
7. 到联邦选举委员会(FDC)登记。
8. 公开声明参选。
9. 募款。
10. 走近平民百姓。 参加初选及党团会议并胜出，或获得多数州代表支持。 参加你所属政党的全国代表大会。
11. 参加大选。
12. 赢得大选。

Gold Reference

1. 总统必须满足在美国本土出生、至少年满三十五岁、在美国至少已居住十四年以上的条件。
2. 打造一个好形象。
3. 明确肢体语言。
4. 丰富自己的经历。
5. 广泛交友。
6. 注册成为正式候选人。
7. 让自己出现在候选人列表上。
8. 设立总统竞选考察委员会。
9. 建立自己的竞选平台。
10. 对时事有一个清晰的认识。
11. 创建竞选纲领。
12. 开始竞选。
13. 磨练辩论技巧。
14. 做好一切准备。

Retrieval Output (Translated)

1. The president must be born in the United States, be at least 35 years old, and have lived in the United States for at least 14 years.
2. Establish a presidential campaign inspection committee.
3. Let yourself appear on the candidate list.
4. Register as an official candidate.
5. Start the election.
6. Create a manifesto.
7. Build your own campaign platform.
8. You must improve your capabilities.
9. Sort.
10. Consider the organization's vision.
11. Teach English.
12. Replace "management" with "leadership".
13. Get an education.
14. Sign a contract.

Generation Output (Translated)

1. Prove that you are a citizen legally born in the United States.
2. You must be at least 35 years old.
3. You must have lived in the United States for at least 14 consecutive years before participating in the election.
4. Learn about the history of American presidents.
5. Discuss carefully with family and supporters.
6. Prepare the campaign research committee.
7. Register with the Federal Election Commission (FDC).
8. Publicly declare to participate in the election.
9. Fundraising.
10. Approach the common people. Participate in primary elections and party caucus meetings and win, or get the support of most state representatives. Attend the national convention of your political party.
11. Participate in the general election.
12. Win the election.

Gold Reference (Translated)

1. The president must be born in the United States, be at least 35 years old, and have lived in the United States for at least 14 years.
2. Create a good image.
3. Use clear body language.
4. Enrich your experience.
5. Make friends widely.
6. Register as an official candidate.
7. Let yourself appear on the candidate list.
8. Establish a presidential campaign inspection committee.
9. Build your own campaign platform.
10. Have a clear understanding of current events.
11. Create a manifesto.
12. Start the election.
13. Hone your debate skills.
14. Be prepared for everything.

Figure 9: The “Run for the US President” script.

Script Name: 弹奏钢琴 (Play the Piano)

Retrieval Output

1. 前面我们讨论的都是C调音阶，它由9个音符组成，从C开始，全部对应的是钢琴的白键。
2. 认识琴的品位。
3. 可以考虑从钢琴入手。
4. 学习键盘。
5. 练习手移动弹奏。
6. 学习和弦。
7. 钢琴或键盘乐器定弦。
8. 学习弹奏乐谱。
9. 获得一架钢琴。
10. 调音管定弦。
11. 使用参照设备定弦。
12. 弹奏一些音符。
13. 学会使用踏板。

Generation Output

1. 你自己要懂琴谱,并且会弹钢琴。 计划一个星期教几节课,每节课时长多少,怎么收费。 决定上课地点。
2. 找学生。
3. 计划上课内容。
4. 上第一节课。
5. 多鼓励你的学生。
6. 加入当地、省际或国际的音乐教师组织。
7. 参加培训。
8. 对于年龄小的学生,你可以建立一个奖励机制。

Gold Reference

1. 获得一架钢琴。
2. 购买配件。
3. 准备正确的姿势。
4. 再次检查你的手部姿势。
5. 学习键盘。
6. 弹奏一些音符。
7. 指法练习。
8. 学习和弦。
9. 学习弹奏乐谱。
10. 单手练习。
11. 练习手移动弹奏。
12. 学会使用踏板。
13. 多练习。

Retrieval Output (Translated)

1. What we discussed earlier is the C scale, which consists of 9 notes. Starting from C, all correspond to the white keys of the piano.
2. Know the capos.
3. Consider starting with the piano.
4. Learn the keyboard.
5. Practice hand movement.
6. Learn chords.
7. Piano or keyboard instrument tuning.
8. Learn to play sheet music.
9. Get a piano.
10. Tuning tube tuning.
11. Use reference equipment to tune.
12. Play some notes.
13. Learn to use the pedal.

Generation Output (Translated)

1. You need to know sheet music yourself and be able to play the piano. Plan how many classes to teach each week, how long each class will be, and how much will be charged. Decide where the class will take place.
2. Find students.
3. Plan class content.
4. Give the first class.
5. Encourage your students.
6. Join a local, inter-provincial or international music teacher organization.
7. Get some training.
8. For younger students, you can establish a reward mechanism.

Gold Reference (Translated)

1. Get a piano.
2. Purchase accessories.
3. Prepare in the correct posture.
4. Check your hand posture again.
5. Learn the keyboard.
6. Play some notes.
7. Fingering practice.
8. Learn chords.
9. Learn to play sheet music.
10. Practice with one hand.
11. Practice hand movement.
12. Learn to use the pedal.
13. practice more.

Figure 10: The “Play the Piano” script.

A Single Example Can Improve Zero-Shot Data Generation

Pavel Burnyshev[1], **Valentin Malykh**[1,2], **Andrey Bout**[1], **Ekaterina Artemova**[1,3], **and Irina Piontkovskaya**[1]

[1]Huawei Noah's Ark Lab, Moscow, Russia
[2]Kazan Federal University, Kazan, Russia
[3]HSE University, Moscow, Russia
{*burnyshev.pavel, malykh.valentin, bout.andrey, artemova.ekaterina, piontkovskaya.irina*}*@huawei.com*

Abstract

Sub-tasks of intent classification, such as robustness to distribution shift, adaptation to specific user groups and personalization, out-of-domain detection, require extensive and flexible datasets for experiments and evaluation. As collecting such datasets is time- and labor-consuming, we propose to use text generation methods to gather datasets. The generator should be trained to generate utterances that belong to the given intent. We explore two approaches to generating task-oriented utterances. In the **zero-shot approach**, the model is trained to generate utterances from seen intents and is further used to generate utterances for intents unseen during training. In the **one-shot approach**, the model is presented with a single utterance from a test intent. We perform a thorough automatic, and human evaluation of the dataset generated utilizing two proposed approaches. Our results reveal that the attributes of the generated data are close to original test sets, collected via crowd-sourcing.

1 Introduction

Training dialogue systems used by virtual assistants in task-oriented applications requires large annotated datasets. The core machine learning task to every dialogue system is *intent detection*, which aims to detect what the intention of the user is. New intents emerge when new applications, supported by the dialogue systems, are launched. However, an extension to new intents may require annotating additional data, which may be time-consuming and costly. What is more, when developing a new dialogue system, one may face the cold start problem if little training data is available. Open sources provide general domain annotated datasets, primarily collected via crowd-sourcing or released from commercial systems, such as Snips NLU benchmark (Coucke et al., 2018). However, it is usually problematic to gather more specific data from any source, including user logs, protected by the privacy policy in real-life settings.

For all these reasons, we suggest a learnable approach to create training data for intent detection. We simulate a real-life situation in which no annotated data but rather only a short description of a new intent is available. To this end, we propose to use methods for zero-shot conditional text generation to generate plausible utterances from intent descriptions. The generated utterances should be in line with the intent's meaning.

Our contributions are:

1. We propose a zero-shot generation method to generate a task-oriented utterance from an intent description;
2. We evaluate the generated utterances and compare them to the original crowd-sourced datasets. The proposed zero-shot method achieves high scores in fluency and diversity as per our human evaluation;
3. We provide experimental evidence of a semantic shift when generating utterances for unseen classes using the zero-shot approach;
4. We apply reinforcement learning for the one-shot generation to eliminate the semantic shift problem. The one-shot approach retains semantic accuracy without sacrificing fluency and diversity.

2 Related work

Conditional language modelling generalizes the task of language modelling. Given some conditioning context z, it assigns probabilities to a sequence of tokens (Mikolov and Zweig, 2012). Machine translation (Sutskever et al., 2014; Cho et al., 2014) and image captioning (You et al., 2016) are seen as typical conditional language modelling tasks. More sophisticated tasks include text abstractive summarization (Nallapati et al., 2017; Narayan et al., 2019) and simplification (Zhang

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 201–211,
Aberdeen, Scotland, UK, 20-24 September 2021.

and Lapata, 2017), generating textual comments to source code (Richardson et al., 2017) and dialogue modelling (Lowe et al., 2017). Structured data may act as a conditioning context as well. Knowledge base (KB) entries (Vougiouklis et al., 2018) or DBPedia triples (Colin et al., 2016) serve as condition to generated plausible factual sentences. Neural models for conditional language modelling rely on encoder-decoder architectures and can be learned both jointly from scratch (Vaswani et al., 2017) or by fine-tuning pre-trained encoder and decoder models (Budzianowski and Vulić, 2019; Lewis et al., 2020).

Zero-shot learning (ZSL) has formed as a recognized training paradigm with neural models becoming more potent in the majority of downstream tasks. In the NLP domain, the ZSL scenario aims at assigning a label to a piece of text based on the label description. The learned classifier becomes able to assign class labels, which were unseen during the training time. The classification task is then reformulated in the form of question answering (Levy et al., 2017) or textual entailment (Yin et al., 2019). Other techniques for ZSL leverage metric learning and make use of capsule networks (Du et al., 2019) and prototyping networks (Yu et al., 2019).

Zero-shot conditional text generation implies that the model is trained in such a way that it can generalize to an unseen condition, for which only a description is provided. A few recent works in this direction show-case dialog generation from unseen domains (Zhao and Eskenazi, 2018) and question generation from KB's from unseen predicates and entity types (Elsahar et al., 2018). CTRL (Keskar et al., 2019), pre-trained on so-called control codes, which can be combined to govern style, content, and surface form, provides for zero-shot generation for unseen codes combinations. PPLM (Dathathri et al., 2019) uses signals, representing the class, e.g., bag-of-words, during inference, and can generate examples with given semantic attributes without pre-training.

Training data generation can be treated as form of data augmentation, a research direction being increasingly in demand. It enlarges datasets for training neural models and help avoid labor-intensive and costly manual annotation. Common techniques for textual data augmentation include back-translation (Sennrich et al., 2016), sampling from latent distributions (Xia et al., 2021), simple heuristics, such as synonym replacement (Wei and Zou, 2019) and oversampling (Chawla et al., 2002). Few-shot text generation has been applied to natural language generation from structured data, such as tables (Chen et al., 2020) and to intent detection data augmentation (Xia et al., 2021). However, these methods are incompatible with ZSL, requiring at least a few labeled examples for the class being augmented. An alternative approach suggests to use a model to generate data for the target class based on task-specific world knowledge (Chen et al., 2017) and linguistic features (Iyyer et al., 2018).

Deep reinforcement learning (RL) methods prove to be effective in a variety of NLP tasks. Early works approach the tasks of machine translation (Grissom II et al., 2014), image captioning (Rennie et al., 2017) and abstractive summarization (Paulus et al., 2017), assessed with not differentiable metrics. (Wu et al., 2021) tries to improve the quality of transformer-derived pre-trained models for generation by leveraging proximal policy optimization. Other applications of deep RL include dialogue modeling (Li et al., 2016b) and open-domain question answering (Wang et al., 2018).

3 Methods

Our main goal is to generate plausible and coherent utterances, which relate to unseen intents, leveraging the description of the intent only. These utterances should clearly express the desired intent. For example, if conditioned on the intent *"delivery from the grocery store"* the model should generate an utterance close to *"Hi! Please bring me milk and eggs from the nearest convenience store"* or similar.

Two scenarios can be used to achieve this goal. In the **zero-shot scenario**, we train the model on a set of seen intents $\mathcal{S}$ to generate utterances. If the generation model generalizes well, the utterances generated for unseen intents $\mathcal{U}$ are diverse and fluent and retain intents' semantics. In the **one-shot scenario**, we utilize one utterance per unseen intent $\mathcal{U}$ to train the generation model and learn the semantics of this particular intent.

3.1 Zero-shot generation

Our model as depicted in Figure 1) aims to generate plausible utterances conditioned on the intent description. We fine-tune the GPT-2 medium model

(Radford et al., 2019) on task-oriented utterances, collected from several NLU benchmarks (see Section 5.1 for more details on the dataset).

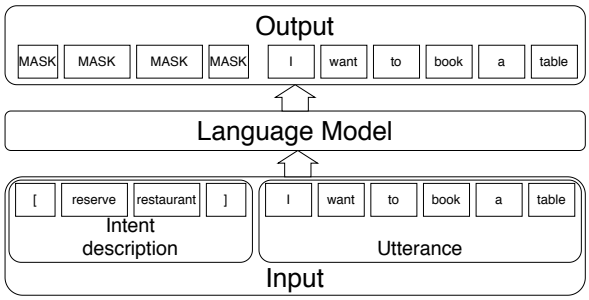


Figure 1: Training setup. The input an intent description and an utterance concatenated, the output is the utterance.

Our approach to fine-tuning the GPT-2 model follows (Budzianowski and Vulić, 2019). Two pieces of information, the intent description and the utterance are concatenated to form the input. More precisely, the input has the following format: *[intent description] utterance*. During the training phase, the model is presented with the output obtained from the input by masking the intent description. The output has the following format: <MASK>, . . ., <MASK> *utterance*. The full list of intents is provided in Table 4 in Appendix.

Such input allows the model to pay attention to intent tokens while generating. The standard language modeling objective, negative log-likelihood loss, is used to train the model:

$$\mathcal{L}(\theta) = -\sum_{i} \sum_{t=1}^{|\mathbf{x}^{(i)}|} \log p_\theta \left(x_t^{(i)} | \text{intent}, x_{<t}^{(i)} \right).$$

We fine-tuned the model for one epoch to avoid over-fitting. Otherwise, the model tends to repeat redundant semantic constructions of the input utterances. At the same time, a bias towards the words from the training set gets formed. The parameters of the training used were set to the following values: batch size equals to 32, learning rate equals to 5e-5, the optimizer chosen is Adam (Kingma and Ba, 2015) with default parameters.

3.2 One-shot Generation

Motivation. The zero-shot approach to conditional generation may degrade or even fail if (i) the intent description is too short to properly reflect the semantics of the intent, (ii) the intent description is ambiguous or contains ambiguous words. Produced utterances may distort the initial meaning of the intent or be meaningless at all. The model may generate an utterance *"Count the number of people in the United States"* for the intent "calculator", or *"Add a book by Shakespeare to the calendar"* for a "book reading" service. Although such examples can be treated not as outliers but rather as real-life whimsical utterances, this is not the desired behavior for the generation model. We address this phenomenon as *Semantic Shift* and provide experimental evidence of it in Section 5.4.

Based on these observations, we hypothesize that the problem could be solved if we provide a single training example to improve models' generalization abilities. A single example can give the model a clue about what the virtual assistant can do with books and which entities our calculator is designed to calculate by gaining better world knowledge. For this purpose, we are moving from the zero-shot to the one-shot setting. We propose a method for improving zero-shot generation by leveraging just one example.

Our approach is inspired by the recent TextGAIL (Wu et al., 2021) approach. It addresses the problem of exposure bias in pre-trained language models and proposes a GAN-like style scheme for fine-tuning GPT-2 to produce appropriate story endings using a reinforcement algorithm. As a reward, TextGAIL uses a discriminator output trained to distinguish real samples from generated samples. As we are limited in using learnable discriminators because of the lack of training data, we propose an objective function based on a similarity score. Our objective function produces utterances, which are close to the reference example. At the same time, it forces the model to generate more diverse and plausible utterances. Table 5 in Appendix provides reference examples used for the one-shot generation method.

Method. After zero-shot fine-tuning, we perform a one-shot model update for each intent separately. We perform several steps of the Proximal Policy Optimization algorithm (Schulman et al., 2017) with the objective function described further.

Reward. Our reward function is based on BERTScore (Zhang et al., 2019), which serves as the measure of contextual similarity between generated sentences and the reference example.

BERTScore correlates better with human judgments than other existing metrics, used to control semantics of generated texts and detect paraphrases. Given a reference and a candidate sentence, we embed them using RoBERTa model (Liu et al., 2019). The BERTScore F1 calculated on top of these embeddings is used as a part of the final reward.

It is not enough to reward the model only for the similarity of the generated utterance to the reference one. If so, the model tends to repeat the reference example and receives the maximal reword. We add the negative sum of frequencies of all n-grams in the utterance to the reward function, forcing the model to generate less frequent sequences.

Given an intent I and a reference example x^I_{ref}, the reward for the sentence x is calculated by the formula:

$$\begin{aligned} R_I(x) &= R_{sim}(x^I_{\text{ref}}, x) + R_{div}(x) \\ R_{sim}(x^I_{\text{ref}}, x) &= \text{BERTScore}(x^I_{\text{ref}}, x) \\ R_{div}(x) &= \sum_{s \in \text{n-grams}(x)} (-\nu_s) \end{aligned}$$

where ν_s is the n-gram frequency, calculated from all the generated utterances inside one batch.

Objective function. First, we plug this reward into standard PPO objective function, getting intent-specific term $L^{\text{policy}}_I(\theta)$. Following the TextGAIL approach, we add $\mathbf{KL}$ divergence with the model without zero-shot fine-tuning to prevent forgetting the information from the pre-trained model. We add an entropy regularizer, making the distribution smoother, which leads to more diverse and fluent sentences. According to our experiments, this term helps avoid similar prefixes for all generated sentences as n-gram reward only does not cope with this issue. The final generator objective for maximization in the one-shot scenario for the intent I can be written as follows:

$$\begin{aligned} L(I;\theta) =& L^{\text{policy}}_I(\theta) + \hat{\mathbb{E}}_t[\beta \mathbf{H}(p_{\theta;I}(\cdot|s_t)) \\ & -\alpha \mathbf{KL}[p_{\theta;I}(\cdot|s_t), q(\cdot|s_t)]], \end{aligned}$$

where s_t is intent description, $p_{\theta;I}$ is the conditional distribution $p_\theta(\cdot|I)$(distribution, derived from model with updates from PPO policy), q is an unconditional LM distribution, calculated by GPT-2 language model without fine-tuning. The entropy and $\mathbf{KL}$ are calculated per each token, while the L^{policy} term is calculated for the whole sentence.

3.3 Decoding strategies

Recent studies show that a properly chosen decoding strategy significantly improves consistency and diversity metrics and human scores of generated samples for multiple generation tasks, such as story generation (Holtzman et al., 2019), open-domain dialogues, and image captioning (Ippolito et al., 2019). However, to the best of our knowledge, no method proved to be a one-size-fits-all one. We perform experiments with several decoding strategies, which improve diversity while preserving the desired meaning. We perform an experimental evaluation of different decoding parameters.

Beam Search, a standard decoding mechanism, keeps the top b partial hypotheses at every time step and eventually chooses the hypothesis that has the overall highest probability.

Random Sampling (top-k) (Fan et al., 2018) greedily samples at each time step one of the top-k most likely tokens in the distribution.

Nucleus Sampling (top-p) (Holtzman et al., 2019) samples from the most likely tokens whose cumulative probability does not exceed p.

Post Decoding Clustering (Ippolito et al., 2019) (i) clusters generated samples using BERT-based similarity and (ii) selects samples with the highest probability from each cluster. It can be combined with any decoding strategy.

4 Performance evaluation

We use several quality metrics to assess the generated data: (i) we use multiple fluency and diversity metrics, (ii) we account for the performance of the classifiers trained on the generated data.

Fluency. We consider fluency dependent upon the number of spelling and grammar mistakes: the utterance is treated as a fluent one if there are no misspellings and no grammar mistakes. We utilize LanguageTool (Miłkowski, 2010), a free and open-source grammar checker, to check spelling and correct grammar mistakes.

Diversity. Following (Ippolito et al., 2019), we consider two types of diversity metrics:

$Dist$-k (Li et al., 2016a) is the total number of distinct k-grams divided by the total number of produced tokens in all of the utterances for an intent;

Ent-k (Zhang et al., 2018) is an entropy of k-grams distribution. This metric takes into consideration that infrequent k-grams contribute more to diversity than frequent ones.

Accuracy. After we obtain a large amount of generated data, we train a RoBERTa-based classifier (Liu et al., 2019) to distinguish between different intents, based on the generated utterances. As usual, we split the generated data into two parts so that the first part is used for training, and the second part serves as the held-out validation set to compute the classification accuracy acc_{clsf}. High acc_{clsf} values mean that the intents are well distinguishable, and the utterances that belong to the same intent are semantically consistent.

Human evaluation We perform two crowdsourcing studies to evaluate the quality of generated utterances, which aim at the evaluation of semantic correctness and fluency.

First, we asked crowd workers to evaluate semantic correctness. We gave crowd workers an utterance and asked them to assign one of the four provided intent descriptions; a correct option was among them (i.e., the one used to generate this very utterance). For the sake of completeness, we added a fifth option, "none of above". We assess the results of this study by two metrics, accuracy and $recall@4$. Accuracy acc_{crowd} measures the number of correct answers, while $recall@4$ measures the number of answers which are different from the last "none of above" option.

Second, we asked crowd workers to evaluate the fluency of generated utterances. Crowd workers were provided with an utterance and were asked to score it on a Likert-type scale from 1 to 5, where (5) means that the utterance sounds natural, (3) means that the utterance contains some errors, (1) means that it is hard or even impossible to understand the utterance. We assess the results of this study by computing the average score.

5 Zero-shot generation experiments

5.1 Data preparation

Data for fine-tuning. We combined two NLU datasets, namely The Schema-Guided Dialogue Dataset (SGD) (Rastogi et al., 2020) and Natural Language Understanding Benchmark (NLU-bench) (Coucke et al., 2018) for the fine-tuning stage. Both datasets have a two-level hierarchical structure: they are organized according to services (in SGD) or scenarios (in NLU-Bench). Each service/scenario contains several intents, typically 2-5 intents per high-level class. For example, the service *Buses_1* is divided into two intents *FindBus* and *BuyBusTickets*.

SGD dataset consists of multi-turn task-oriented dialogues between user and system; each user utterance is labeled by service and intent. We adopted only those utterances from each dialog in which a new intent arose, which means the user clearly announced a new intention. This is a common technique to remove sentences that do not express any intents. As a result, we got three utterances per dialog on average.

As NLU-Bench consists of user utterances, each marked up with a scenario and intent label, we used it without filtering. Summary statistics of the dataset used is provided in Table 1.

	SGD	NLU-bench	Total
No. of utterances	49986	25607	75593
No. of services	32	18	50
No. of intents	67	68	135
Total tokens	∼550k	∼170k	∼720k
Unique tokens	∼10.8k	∼8.3k	∼17.4k

Table 1: The total number of utterances, intents, services and words across datasets and final statistics of our fine-tuning data.

Intent set for generation. For the evaluation of our generation methods, we created a set of 38 services and 105 intents[1] covering the most common requirements of a typical user of a modern dialogue system. The set includes services dedicated to browsing the Internet, adjusting mobile device settings, searching for vehicles, and others. To adopt a zero-shot setup, we split the data into train and test sets in the following way. Some of the services are unseen ($s \in \mathcal{U}$), i.e., are present in the test set only. There are no seen services in the train set related to unseen services. The rest of the services are seen, i.e., present in both train and test set ($s \in \mathcal{S}$), but different intents put in train and test sets. For example, *Flight* services are present in the train data and *Plane* service is

[1]The full list of services and intents in both sets presented in the Appendix

Decoding strategy	Automated metrics			Human evaluation		
	acc_{clsf}	$Dist$-4	Ent-4	acc_{crowd}	$recall$@4	Fluency score
Zero-shot generation						
Random Sampling ($b = 4$)	0.82	**0.50**	**6.20**	0.63	0.87	4.77
Nucleus Sampling ($p = 0.6$) + PDC	0.82	0.40	5.77	0.68	0.85	**4.95**
Beam Search ($b = 3$) + PDC	0.85	0.22	4.92	0.67	0.85	4.88
Beam Search ($b = 3$)	0.88	0.15	4.76	0.60	0.80	4.76
Nucleus Sampling ($p = 0.4$)	0.89	0.25	4.95	0.72	0.90	4.81
One-shot generation						
Nucleus Sampling ($p = 0.4$)	**0.94**	0.39	5.88	**0.78**	**0.91**	4.86

Table 2: Decoding strategies for zero-shot and one-shot generation. PDC stands for Post Decoding Clustering.

	acc_{crowd}	$recall$@4	$Dist$-4	Ent-4
SGD+NLU-bench	0.83	0.95	0.53	5.92

Table 3: Evaluation of the test dataset, created by merging and re-splitting two datasets under consideration.

used in the test set; from *Music* services, intents *Lookup song* and *Play song* were used for training, and *Create playlist* and *Turn on music* for a testing. To form the intent description for fine-tuning and generation, we join service and intent labels.

5.2 Evaluation

We generated 100 examples per intent using different decoding strategies and their parameters. For the more detailed evaluation, we picked up the generation methods of different decoding strategies that achieved good scores ($acc_{clsf} > 80\%$ and Ent-4 > 4). For these utterances, we performed a human evaluation of semantic correctness and diversity; Table 2 compares the decoding strategies according to various quality metrics. For a more detailed evaluation of decoding strategies, see Table 2 in Appendix.

To compare the diversity of human-generated utterances to our generated utterances, we evaluate the fine-tuning dataset with Ent-4 and $Dist$-4 metrics. The semantics of generated data is assessed by acc_{crowd} and $recall$@4. We present metrics for this dataset in Table 3.

5.3 Analysis and model comparison

Fluency. Spell checking results reveal the following issues of the generated utterances. The major issues are related to casing: an utterance may start in lower case, the first-person singular personal pronoun "I" is frequently generated in lower case, too. Punctuation issues include missing quotes, question marks, periods, or repeated punctuation marks. Common mistakes are omitting of a hyphen in the word *"Wi-Fi"* and *"e-mail"* and confusing definite and indefinite articles, as well as confusing *"a"/"an"*. These issues are more or less natural to humans and thus do not prevent further use of generated utterances. The only unnatural issues found by LanguageTool are phrase repetition in small numbers (4 errors of this type per 10000 utterances). For examples of fluency issues in generated data, see Table 1 in Appendix.

Diversity. Table 4 shows examples of the phrases generated by means of different decoding strategies, conditioning on the intent *Show message*, along with diversity metrics, $Dist$ and Ent. Higher Ent and $Dist$ scores indeed correspond to a more diverse decoding strategy. At the same time, extremely high diversity may generate utterances unrelated to the intent, expressing non-clear meaning and lack of common sense.

Diversity / Accuracy trade-off. Figure 2 shows the trade-off between the diversity (Ent-4) and the accuracy (acc_{clsf}) of the generated data.

Every point corresponds to sentences generated using different zero-shot strategies. The human level stands for the diversity and accuracy metrics computed for the test set as is. The beam search scores are mainly in the top-left corner of the plane, leading to high accuracy and low diversity values.

Beam Search (3) $Ent\text{-}4 = 4.26$	Random Sampling (3) $Ent\text{-}4 = 5.93$	Nucleus Sampling (0.98) $Ent\text{-}4 = 6.86$
i need to know what's going on with my phone	i want to see my messages in the phone book	show me a message from jean lee for my favorite apple company
i want you show me the message from my phone	show me my most recent messages from my phone number	how can you tell me mike with the message
i want you show me my messages on my phone	show me the messages from the device i was using	could you check to see if my friends are in a group that is gossiping
i want you to show my messages on my smart phone	show me the message from my friend jane that i sent to her	list all messages in my bbq menu from ausy
i want to read a new message from my friend	can you please show me the messages from my phone	just turn on the smart mute this monday night

Table 4: Utterances, generated by different decoding strategies and the diversity scores of the decoding strategies.

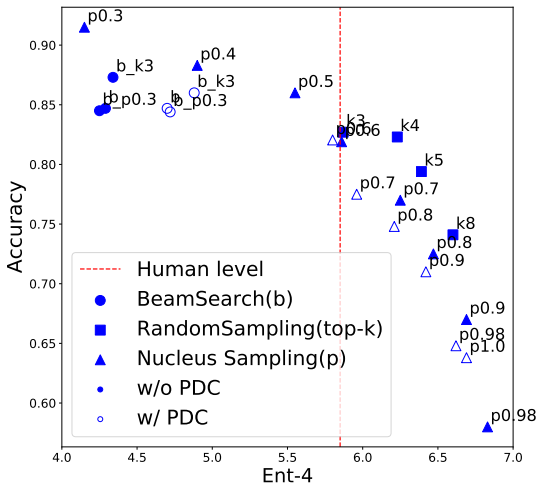


Figure 2: The trade-off between diversity (Ent-4) and accuracy.

Top-k Random Sampling strategy does not achieve the highest levels of accuracy. Nucleus Sampling can generate datasets with a large range of diversity and accuracy scores, depending on the chosen parameter. Post-decoding clustering increases diversity for low-diverse decoding strategies and decreases it for high-diverse ones, moving the generator closer to the human level.

Two ways to assess accuracy. Table 2 shows that there is no clear correspondence between automated accuracy acc_{clsf} and human accuracy acc_{crowd}. Therefore acc_{clsf} cannot serve as the final measure for the semantic consistency of the generator. The *Semantic shift* problem cannot be captured by the automated accuracy acc_{clsf}: the model generates examples which are consistent inside each class, and classes are well-separated, but the generated examples do not correspond well to the intent descriptions.

5.4 Semantic shift problem

The semantic consistency is crucial: how well do the generated utterances correspond to the intent description? In most cases, zero-shot generation is quite reliable: $acc_{crowd} > 0.8$ for 57% of intents, $recall@4 > 0.9$ for 72% of intents. However, generated utterances are distinguishable from other classes for some intents, but they do not completely correspond to the intent description. Several generated utterances below illustrate this issue.

Intent: *Buy train tickets*
Utterance: I want to buy a bus ticket. I want to leave on the 12th of this month.
Intent: *Put default wallpapers*
Utterance: Put the default wallpaper for the bedroom. I want to see it on the wall.
Intent: *Calculator Find sum*
Utterance: I need to find a calculator. I need to know the value of one dollar.

For example, The bias in the fine-tuning data causes this issue. For example, travel-related intents mainly correspond to bus travel. So the model confuses buses and trains. In other cases, the model gets wrong the intent description due to the lack of world knowledge. E. g. the generated phrases for *Wallpaper* may be related to wallpapers in a house; utterances for *Calculator* may be related to finding some numbers like the average price of houses in the area.

Intent description and reference examples	Undesirable meaning	Zero-shot	One-shot
Intent description Train Buy train ticket **Reference** Make a purchase of the train ticket, not bus. Buy a train ticket for a specific date to some location	**Meaning** Get bus ticket **Example** I need a bus to go there. I need to leave on the 3rd of this month.	97	23
Intent description Wallpapers Put default wallpaper **Reference** Change the background picture of the device display to the default one. Replace current background on the device with the default one	**Meaning** Put new wall cover in a house **Example** I want to put the wallpaper for my bedroom on the wall.	74	1
Intent description Calculator Find sum **Reference** Compute, calculate the sum of the given numbers. Open the calculator and compute the sum of the following numbers	**Meaning** Find some amount of money **Example** I need to find the average price of a house.	57	0

Table 5: Evaluation of semantic shift reduction by one-shot generation. The first column contains intent description and reference utterances used for one-shot generation. The second column shows examples of typical undesirable meaning. The last two columns show the percentage of examples with given incorrect meaning among 100 generated utterances by zero-shot and one-shot generation. Nucleus sampling ($p = 0.4$) is used for both methods.

6 One-shot generation experiments

Based on human evaluation of zero-shot generated data, we select Nucleus Sampling ($p = 0.4$) as the best decoding strategy and apply it further in the one-shot scenario. Indeed, Table 2 confirms that the one-shot generation improves all evaluation metrics, both human and automated. The resulting one-shot utterances are more fluent than zero-shot utterances. The classifier trained on one-shot utterances has higher accuracy values when compared to the one trained on zero-shot utterances.

At the same time, one-shot generation restricts the semantics of the generated utterances and reduces the semantic shift. To illustrate, how the problem of semantic shift diminishes, we study several cases where the zero-shot model tends to generate utterances with undesirable meaning (see Section 5.4): **bus** instead of **train**; **wallpaper** as a *wall cover* instead of *background picture*; **sum** as *amount of money* instead of *number*. Table 5 shows that after one-shot fine-tuning, the number of utterances with undesirable meaning becomes drastically lower; for more examples, see Table 3 in Appendix.

7 Conclusion

In this paper, we have introduced zero-shot and one-shot methods for generating utterances from intent descriptions. We ensure the high quality of the generated dataset by a range of different measures for diversity, fluency, and semantic correctness, including a crowd-sourcing study. We show that the one-shot generation outperforms the zero-shot one based on all metrics considered. Using only a single utterance for an unseen intent to fine-tune the model increases diversity and fluency. Moreover, fine-tuning on a single utterance diminishes the semantic shift problem and helps the model gain better world knowledge.

Virtual assistants in real-life setup should be highly adaptive. In some tasks, we need much more data than is currently available: exploring model robustness to distribution change, finding the best architecture, dealing with a fast-growing set of intents (the number of intents could be thousands). If the intents to support come from different providers, they pose diverse semantics, style, and noises. Adaptation to different user groups and individual users, having different intent usage distribution, is another crucial problem. We need large-scale and flexible datasets to approach these tasks, which can hardly be collected via crowd-sourcing from external sources.

Zero- or one-shot generation is an appealing technique. The model obtains the background knowledge about the world and the domain during pre-training. Next, only small amounts of data are

needed to fine-tune the model. State-of-the-art pre-trained language models, fine-tuned in a zero- or one-shot fashion, generate fluent and diverse phrases close to real-life utterances. The meaning of the intent and essential details, such as book titles, movie genres, expression of speech acts, or emoticons, are preserved. What is more, manipulating a decoding strategy makes it possible to balance the generated utterances' diversity, semantic consistency, and correctness.

Our future work directions include assessing the downstream performance of proposed generation methods for an end-user application and evaluating slot-filling performance. The proposed approach can be tested to generate utterances specific to interest groups.

Acknowledgements

Ekaterina Artemova is partially supported by the framework of the HSE University Basic Research Program.

References

Paweł Budzianowski and Ivan Vulić. 2019. Hello, it's gpt-2-how can i help you? towards the use of pretrained language models for task-oriented dialogue systems. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 15–22.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically Labeled Data Generation for Large Scale Event Extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–419.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2020. Few-shot nlg with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Emilie Colin, Claire Gardent, Yassine M'rabet, Shashi Narayan, and Laura Perez-Beltrachini. 2016. The webnlg challenge: Generating text from dbpedia data. In *Proceedings of the 9th International Natural Language Generation conference*, pages 163–167.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. SNIPS Voice Platform: an Embedded Spoken Language Understanding System for Private-by-Design Voice Interfaces. *arXiv preprint arXiv:1805.10190*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. In *International Conference on Learning Representations*.

Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, Jianxin Liao, Chun Wang, and Bing Ma. 2019. Investigating Capsule Network and Semantic Feature on Hyperplanes for Text Classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 456–465.

Hady Elsahar, Christophe Gravier, and Frederique Laforest. 2018. Zero-shot question generation from knowledge graphs for unseen predicates and entity types. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 218–228.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical Neural Story Generation. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't Until the Final Verb Wait: Reinforcement learning for Simultaneous Machine Translation. In *Proceedings of the 2014 Conference on empirical methods in natural language processing (EMNLP)*, pages 1342–1352.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*.

Daphne Ippolito, Reno Kriz, Maria Kustikova, João Sedoc, and Chris Callison-Burch. 2019. Comparison of Diverse Decoding Methods from Conditional Language Models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3752–3762.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In *Proceedings of NAACL-HLT*, pages 1875–1885.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL:

A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A Diversity-Promoting Objective Function for Neural Conversation Models. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep Reinforcement Learning for Dialogue Generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

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

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65.

Tomas Mikolov and Geoffrey Zweig. 2012. Context Dependent Recurrent Neural Network Language Model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239. IEEE.

Marcin Miłkowski. 2010. Developing an Open-source, Rule-based Proofreading Tool. *Software: Practice and Experience*, 40(7):543–566.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2019. What is this article about? extreme summarization with topic-aware convolutional neural networks. *Journal of Artificial Intelligence Research*, 66:243–278.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A Deep Reinforced Model for Abstractive Summarization. *arXiv preprint arXiv:1705.04304*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical Sequence Training for Image Captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.

Kyle Richardson, Sina Zarrieß, and Jonas Kuhn. 2017. The code2text challenge: Text generation in source libraries. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 115–119.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27:3104–3112.

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

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. 2018. Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 52:1–15.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced Ranker-Reader for Open-domain Question Answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jason Wei and Kai Zou. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint*

Conference on Natural Language Processing (EMNLP-IJCNLP), pages 6383–6389.

Qingyang Wu, Lei Li, and Zhou Yu. 2021. Textgail: Generative adversarial imitation learning for text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14067–14075.

C Xia, C Xiong, and PS Yu. 2021. Pseudo siamese network for few-shot intent generation. In *ACM SIGIR*.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3905–3914.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659.

Yunlong Yu, Zhong Ji, Zhongfei Zhang, and Jungong Han. 2019. Episode-based prototype generating network for zero-shot learning.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization. *Advances in Neural Information Processing Systems*.

Tiancheng Zhao and Maxine Eskenazi. 2018. Zero-shot dialog generation with cross-domain latent actions. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 1–10.

SAPPHIRE: Approaches for Enhanced Concept-to-Text Generation

Steven Y. Feng, Jessica Huynh, Chaitanya Narisetty, Eduard Hovy, Varun Gangal
Language Technologies Institute
Carnegie Mellon University
{syfeng,jhuynh,cnariset,hovy,vgangal}@cs.cmu.edu

Abstract

We motivate and propose a suite of simple but effective improvements for concept-to-text generation called SAPPHIRE: Set Augmentation and Post-hoc PHrase Infilling and REcombination. We demonstrate their effectiveness on generative commonsense reasoning, a.k.a. the *CommonGen* task, through experiments using both BART and T5 models. Through extensive automatic and human evaluation, we show that SAPPHIRE noticeably improves model performance. An in-depth qualitative analysis illustrates that SAPPHIRE effectively addresses many issues of the baseline model generations, including lack of commonsense, insufficient specificity, and poor fluency.

1 Introduction

There has been increasing interest in constrained text generation tasks which involve constructing natural language outputs under certain preconditions, such as particular words that must appear in the output sentences. A related area of work is data-to-text natural language generation (NLG), which requires generating natural language descriptions of structured or semi-structured data inputs. Many constrained text generation and NLG tasks share commonalities, one of which is their task formulation: a set of inputs must be converted into natural language sentences. This set of inputs can be, in many cases, thought of as *concepts*, e.g. higher-level words or structures that play an important role in the generated text.

With the increased popularity of Transformer-based models and their application to many NLP tasks, performance on many text generation tasks has improved considerably. Much progress in recent years has been from the investigation of model improvements, such as larger and more effectively pretrained language generation models. However, are there simple and effective approaches to improving performance on these tasks that can come from the data itself? Further, can we potentially use the outputs of these models themselves to further improve their task performance - a "self-introspection" of sorts?

In this paper, we show that the answer is yes. We propose a suite of simple but effective improvements for concept-to-text generation called SAPPHIRE: Set Augmentation and Post-hoc PHrase Infilling and REcombination. Specifically, SAPPHIRE is composed of two major approaches: 1) the augmentation of input concept sets (§4.1), 2) the recombination of phrases extracted from baseline generations into more fluent and logical text (§4.2). These are mainly model-agnostic improvements that rely on the data itself and the model's own initial generations, respectively.[1]

We focus on generative commonsense reasoning, or CommonGen (Lin et al., 2020), which involves generating logical sentences describing an everyday scenario from a set of concepts, which in this case are individual words that must be represented in the output text in some form. CommonGen is a challenging instance of constrained text generation that assesses 1) relational reasoning abilities using commonsense knowledge, and 2) compositional generalization capabilities to piece together concept combinations. Further, CommonGen's task formulation and evaluation methodology are quite broadly applicable and encompassing, making it a good benchmark for general constrained text generation capability. Further, this is an opportune moment to investigate this task as commonsense ability of NLP models, particularly for generation, has received increasing community attention through works like COMET (Bosselut et al., 2019).

We perform experiments on varying sizes of two

[1] Code at https://github.com/styfeng/SAPPHIRE

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 212–225,
Aberdeen, Scotland, UK, 20-24 September 2021.

Dataset Stats	Train$_{CG}$	Dev$_O$	Test$_O$	Dev$_{CG}$	Test$_{CG}$
# concept sets	32,651	993	1,497	240	360
size = 3	25,020	493	-	120	-
size = 4	4,240	250	747	60	180
size = 5	3,391	250	750	60	180
# sentences	67,389	4,018	7,644	984	1583

Table 1: CommonGen dataset statistics.

state-of-the-art Transformer-based language generation models: BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). We first conduct an extensive correlation study (§3.1) and qualitative analysis (§3.2) of these models' generations after simply training on CommonGen. We find that performance is positively correlated with concept set size, motivating concept set augmentation. We also find that generations contain issues related to commonsense and fluency which can possibly be addressed through piecing the texts back together in different ways, motivating phrase recombination.

Fleshing out our first intuition - we devise two methods to augment concepts from references during training through extracted keywords (§4.1.1) and attention matrices (§4.1.2). For the phrase recombination intuition, we propose two realizations based on a new training stage (§4.2.1) and masked infilling (§4.2.2). Finally, through comprehensive evaluation (§6), we show how the SAPPHIRE suite drives up model performance across metrics, besides addressing aforementioned baseline deficiencies on commonsense, specificity, and fluency.

2 Dataset, Models, and Metrics

2.1 CommonGen Dataset

The CommonGen dataset is split into train, dev, and test splits, covering a total of 35,141 concept sets and 79,051 sentences. The concept sets range from 3 keywords to 5 keywords long. As the original test set is hidden, we split the provided dev set into a new dev and test split for the majority of our experiments while keeping the training split untouched. Note that we also evaluate our SAPPHIRE models on the original test set with help from the CommonGen authors (see §6.1). We will henceforth refer to these new splits as train$_{CG}$, dev$_{CG}$, and test$_{CG}$, and the original dev and test splits as dev$_O$ and test$_O$. The statistics of our new splits compared to the originals can be found in Table 1. We attempt to keep the relative sizes of the new dev and test splits and the distribution of concept set sizes within each split similar to the originals.

Model\Metrics	BLEU-4	CIDEr	SPICE
Reported BART-large	27.50	14.12	30.00
Reported T5-base	18.00	9.73	23.40
Reported T5-Large	30.60	15.84	31.80
Our BART-base	28.30	15.07	30.35
Our BART-large	**30.20**	**15.72**	**31.20**
Our T5-base	**31.00**	**16.37**	**32.05**
Our T5-large	**33.60**	**17.02**	**33.45**

Table 2: Performance of our re-implemented CommonGen models on dev$_O$ compared to a subset of original numbers reported in Lin et al. (2020). For our models, results are averaged over two seeds. The original authors did not experiment with BART-base. Bold indicates where we match or exceed the reported metric. See §2.3 for explanations of the metrics and Appendix B for a full metric comparison table.

2.2 Models: T5 and BART

We perform experiments using pretrained language generators, specifically BART and T5 (both base and large versions). BART (Lewis et al., 2020) is a Transformer-based seq2seq model trained as a denoising autoencoder to reconstruct original text from noised text. T5 (Raffel et al., 2020) is another seq2seq Transformer with strong multitask pretraining. We use their HuggingFace codebases.

We train two seeded instances of each model on train$_{CG}$, evaluating their performance on dev$_O$, and comparing our numbers to those reported in Lin et al. (2020) to benchmark our implementations. These essentially serve as the four baseline models for our ensuing experiments. We follow the hyperparameters from Lin et al. (2020), choose the epoch reaching highest ROUGE-2 on the dev split, and use beam search for decoding.[2] From Table 2, we see that our re-implemented models match or exceed the original reported results on most metrics across different models.

2.3 Evaluation Metrics

For our experiments, we use a gamut of automatic evaluation metrics. These include those used by Lin et al. (2020), such as BLEU (Papineni et al., 2002), CIDEr (Vedantam et al., 2015), SPICE (Anderson et al., 2016), and Coverage (Cov). Barring Cov, these metrics measure the similarity between generated text and human references. Cov measures the average % of input concepts covered by the generated text. We also introduce BERTScore (Zhang et al., 2020), which measures token-by-token BERT (Devlin et al., 2019) embeddings similarity. It also measures the similarity between the generated text and human references, but on a more semantic (rather than surface token) level. When

[2]See Appendix A for further details.

reporting BERTScore, we multiply by 100. For all metrics, higher corresponds to better performance.

3 Initial Analysis

3.1 Correlation Study

We begin by conducting an analysis of the four baselines implemented and discussed in §2.2, which we refer to henceforth as BART-base-BL, BART-large-BL, T5-base-BL, and T5-large-BL. One aspect we were interested in is whether the number of input concepts affects the quality of generated text. We conduct a comprehensive correlation study of the performance of the four baselines on dev_O w.r.t. the number of input concepts.

As seen from Table 3, the majority of the metrics are positively correlated with concept set size across the models. ROUGE-L, CIDEr, and SPICE have small correlations that are mainly statistically insignificant, demonstrating that they are likely uncorrelated with concept set size. Coverage is strongly negatively correlated, showing that there is a higher probability of concepts missing from the generated text as concept set size increases.

There are two major takeaways from this. Firstly, increased concept set size results in greater overall performance. Secondly, models have difficulty with coverage given increased concept set size. This motivates our first set of improvements, which involves augmenting the concept sets with additional words in hopes of 1) increasing performance of the models and 2) improving their coverage, as we hope that training with more input concepts will help models learn to better cover them in the generated text. This is discussed more in §4.1.

3.2 Qualitative Analysis

We conduct a qualitative analysis of the baseline model outputs. We observe that several outputs are more like phrases than full coherent sentences, e.g. *"body of water on a raft"*. Some generated texts are also missing important words, e.g. *"A listening music and dancing in a dark room"* is clearly missing a noun before *listening*. A large portion of generated texts are quite generic and bland, e.g. *"Someone sits and listens to someone talk"*, while more detailed and specific statements are present in the human references. This can be seen as an instance of the noted *"dull response" problem* faced by generation models (Du and Black, 2019; Li et al., 2016), where they prefer safe, short, and frequent responses independent of the input.

Another issue is the way sentences are pieced together. Certain phrases in the outputs are either joined in the wrong order or with incorrect connectors, leading to sentences that appear to lack commonsense. For example, *"body of water on a raft"* is illogical, and the phrases *"body of water"* and *"a raft"* are pieced together incorrectly. Example corrections include *"body of water carrying a raft"* and *"a raft on a body of water"*. The first changes the adverb *on* joining them to the verb *carrying*, and the second pieces them together in the opposite order. A similar issue occurs with the {*horse, carriage, draw*} example in Table 4.

Some major takeaways are that many generations are: 1) phrases rather than full sentences and 2) poorly pieced together and lack fluency and logic compared to human references. This motivates our second set of improvements, which involves recombining extracted phrases from baseline generations into hopefully more fluent and logical sentences. This is discussed more in §4.2.

4 SAPPHIRE Methodology

4.1 Concept Set Augmentation

The first set of improvements is concept set augmentation, which involves augmenting the input concept sets. We try augmentation using up to 1 to 5 additional words, and train-time augmentation both with and without test-time augmentation. We observed that test-time augmentation resulted in inconsistent results that were not as effective, and stick with train-time only augmentation. During training, rather than feeding in the original concept sets as inputs, we instead feed in these augmented concept sets which consist of more words. The expected outputs are the same human references. During test-time, we simply feed in the original concept sets (without augmentation) as inputs.

4.1.1 Keyword-based Augmentation

The first type of augmentation we try is keyword-based, or *Kw-aug*. We augment the train_{CG} concept sets with keywords extracted from the human references using KeyBERT[3] (Grootendorst, 2020). We calculate the average semantic similarity (using cosine similarity of BERT embeddings) of the candidate keywords to the original concept set. At each stage of augmentation, we add the remaining can-

[3] https://github.com/MaartenGr/KeyBERT

	BART-base			BART-large			T5-base			T5-large		
Correlation	PCC	ρ	τ	PCC	ρ	τ	PCC	ρ	τ	PCC	ρ	τ
ROUGE-1	0.08	0.09	0.07	0.10	0.12	0.09	0.04	0.05	0.04	0.10	0.11	0.09
ROUGE-2	0.05	0.08	0.07	0.05	0.10	0.07	0.03	0.07	0.05	0.06	0.09	0.07
ROUGE-L	0.00*	0.01*	0.01*	0.00*	0.02*	0.01*	-0.03	-0.01*	-0.01*	0.02*	0.04	0.03
BLEU-1	0.08	0.08	0.06	0.14	0.14	0.11	0.00*	0.03*	0.02*	0.08	0.11	0.09
BLEU-2	0.06	0.06	0.04	0.11	0.11	0.08	0.03*	0.04*	0.03*	0.09	0.10	0.07
BLEU-3	0.08	0.06	0.05	0.09	0.09	0.06	0.04*	0.03*	0.02*	0.09	0.08	0.06
BLEU-4	0.05	0.05	0.04	0.05	0.07	0.05	0.04*	0.02*	0.02*	0.08	0.08	0.06
METEOR	0.05	0.08	0.06	0.06	0.09	0.07	0.02*	0.04	0.03	0.06	0.08	0.06
CIDEr	-0.02*	-0.03*	-0.02*	0.01*	0.02*	0.02*	-0.08	-0.10	-0.07	0.00*	0.00*	0.00*
SPICE	-0.02*	-0.01*	-0.01*	0.01*	0.02*	0.01*	-0.02*	-0.02*	-0.02*	0.02*	0.03*	0.02*
BERTScore	0.04	0.03	0.02	0.06	0.06	0.05	0.04	0.03	0.02	0.05	0.04	0.03
Coverage	-0.26	-0.31	-0.27	-0.07	-0.13	-0.11	-0.38	-0.42	-0.37	-0.26	-0.31	-0.28

Table 3: Correlations on dev_O between concept set size and evaluation metrics for our four baseline models (over the results from both seeds); values marked with * are statistically insignificant. PCC refers to Pearson correlation coefficient, ρ to Spearman's rank correlation coefficient, and τ to Kendall rank correlation coefficient.

Concept Set	Baseline Generation	Human Reference
{horse, carriage, draw}	horse drawn in a carriage	The carriage is drawn by the horse.
{fish, catch, pole}	fish caught on a pole	The man used a fishing pole to catch fish.
{listen, talk, sit}	Someone sits and listens to someone talk.	The man told the boy to sit down and listen to him talk.
{bathtub, bath, dog, give}	A dog giving a bath in a bathtub.	The teenager made a big mess in the bathtub giving her dog a bath.

Table 4: Example generations from our baseline models versus human references.

Method	Original Concept Set	Added Words
Kw-aug	{match, stadium, watch}	{soccer, league, fans}
Kw-aug	{family, time, spend}	{holidays}
Kw-aug	{head, skier, slope}	{cabin}
Att-aug	{boat, lake, drive}	{fisherman}
Att-aug	{family, time, spend}	{at, holidays}
Att-aug	{player, match, look}	{tennis, on, during}

Table 5: Example train_{CG} concept set augmentations.

didate with the highest similarity.[4] Some augmentation examples can be found in Table 5. We train our BART and T5 models using these augmented sets, and call the resulting models BART-base-KW, BART-large-KW, T5-base-KW, and T5-large-KW.

4.1.2 Attention-based Augmentation

We also try attention-based augmentation, or *Att-aug*. We augment the train_{CG} concept sets with the words that have been most attended upon in aggregate by the other words in the human references. We pass the reference texts through BERT and return the attention weights at the last layer. At each stage of augmentation, we add the remaining candidate word with the highest attention. Adding the least attended words would not be effective as many are words with little meaning (e.g. simple articles such as *"a"* and "the"). Some augmentation examples can be found in Table 5. We train our BART and T5 models using these augmented sets, and call the resulting models BART-base-Att, BART-large-Att, T5-base-Att, and T5-large-Att.

[4]We also tried using the least semantically similar keywords, but results were noticeably worse.

4.2 Phrase Recombination

The second set of improvements is phrase recombination, which involves breaking down sentences into phrases and reconstructing them into new sentences which are hopefully more logical and coherent. For training, we use YAKE (Campos et al., 2018) to break down the train_{CG} human references into phrases of up to 2, 3, and 5 n-grams long, and ensure extracted phrases have as little overlap as possible. During validation and testing, since we assume no access to ground-truth human references, we instead use YAKE to extract keyphrases from our baseline model generations.

We ignore extracted 1-grams as this approach focuses on phrases. We find words from the original concept set which are not covered by our extracted keyphrases and include them to ensure that coverage is maintained. Essentially, we form a new concept set which can also consist of phrases. Some examples can be found in Table 6.

4.2.1 Phrase-to-text (P2T)

To piece the phrases back together, we try phrase-to-text (P2T) generation by training BART and T5 to generate full sentences given our new input sets, and call these models BART-base-P2T, BART-large-P2T, T5-base-P2T, and T5-large-P2T. During

Original Text	Extracted Keyphrases	New Input Concept Set
A dog wags his tail at the boy.	dog wags his tail	{dog wags his tail}
hanging a painting on a wall at home	hanging a painting	{hanging a painting, wall}
a herd of many sheep crowded together in a stable waiting to be dipped for ticks and other pests	herd of many sheep crowded	{herd of many sheep crowded, dip, waiting}
a soldier takes a knee while providing security during a patrol outside of the village.	knee while providing security, patrol outside of the village	{knee while providing security, patrol outside of the village, take}

Table 6: Example keyphrases (up to 5-grams) extracted using YAKE from human-written training references.

training, we choose a single random permutation of each training input set (consisting of extracted keyphrases from the human references), with the elements separated by $<s>$, and the human references as the outputs. This is in order for the models to learn to be order-agnostic, which is important as one desired property of phrase recombination is the ability to combine phrases in different orders, as motivated by the qualitative analysis in §3.2. During inference or test-time, we feed in a single random permutation of each test-time input set, consisting of extracted keyphrases from the corresponding baseline model's outputs.

4.2.2 Mask Infilling (MI)

This method interpolates text between test-time input set elements with no training required. For example, given a test-time input set $\{c_1,c_2\}$, we feed in *"<mask> c_1 <mask> c_2 <mask>"* and *"<mask> c_2 <mask> c_1 <mask>"* to an MI model to fill the <mask> tokens with text. We use BART-base and BART-large for MI, and call the approaches BART-base-MI and BART-large-MI, respectively. We use BART-base-MI on input sets consisting of extracted keyphrases from BART-base-BL and T5-base-BL, and BART-large-MI on input sets consisting of extracted keyphrases from BART-large-BL and T5-large-BL. We also try MI on the original concept sets (with no phrases).

One difficulty is determining the right input set permutation. Many contain $\geq$5 elements (meaning $\geq$5!=120 permutations), making exhaustive MI infeasible. Order of elements for infilling can result in vastly different outputs (see §6.3), as certain orders are more natural. Humans perform their own intuitive reordering of given inputs when writing, and the baselines and other approaches (e.g. Kw-aug, P2T) learn to mainly be order agnostic.

We use perplexity (PPL) from GPT-2 (Radford et al., 2019) to pick the *"best"* permutations for MI. We feed up to 120 permutations of each input set (with elements separated by spaces) to GPT-2 to extract their PPL, and keep the 10 with lowest PPL per example. This is not a perfect approach, but is likely better than random sampling. For each example, we perform MI on these ten permutations, and select the output with lowest GPT-2 PPL.

We found BART-large-MI outputs contain URLs, news agency names in brackets, etc. Hence, we post-process before output selection and evaluation. BART-base-MI does not do this. One possible explanation is that BART-large may have been pretrained on more social media and news data.

5 Experiments

5.1 Model Training and Selection

For training Kw-aug, Att-aug, and P2T models, we follow baseline hyperparameters, barring learning rate (LR) which is tuned per-method. We train two seeds per model. See Appendix A for more.

For each model, we choose the epoch corresponding to highest ROUGE-2 on the dev split, and use beam search for decoding. The dev and test splits are different. For Kw-aug and Att-aug models, the splits are simply dev_{CG} and test_{CG} (or test_{O}), as we do not perform test-time augmentation. For P2T, the splits are dev_{CG} and test_{CG} (or test_{O}) but with the input sets replaced with new ones that include keyphrases extracted from the corresponding baseline model's outputs.

The number of words to augment for Kw-aug and Att-aug (from 1 to 5) and maximum n-gram length of extracted keyphrases for P2T (2, 3, or 5) are hyperparameters. While we train separate versions of each model corresponding to different values of these, the final chosen model per method and model combination (such as BART-base-KW) is the one corresponding to the hyperparameter value that performs best on the dev split when averaged over both seeds. For MI, which involves no training, we select the variation (MI on the original concept set or new input sets with keyphrases up to 2, 3, or 5 n-grams) per model that performs best on the dev split, and only perform infilling using extracted keyphrases from the first seed baseline generations. These are the selected models we report the test_{CG} and test_{O} results of in §6.

5.2 Human Evaluation

We ask annotators to evaluate 48 test$_{CG}$ examples from the human references, baseline outputs, and various method (excluding MI) outputs for BART-large and T5-base. We choose these two as they cover both model types and sizes, and exclude MI as it performs noticeably worse on the automatic evaluation (see §6.1). See Appendix §C for more.

The annotators evaluate fluency and commonsense of the texts on 1-5 scales. Fluency, also known as naturalness, is a measure of how human-like a text is. Commonsense is the plausibility of the events described. We do not evaluate coverage as automatic metrics suffice; coverage is more objective compared to fluency and commonsense.

6 Results and Analysis

Automatic evaluation results on test$_{CG}$ can be found in Tables 7, 8, 9, 10, and results on test$_{O}$ in Table 12. Human evaluation results on test$_{CG}$ can be found in Table 13. Single keyword augmentation performs best for Kw-aug across models. Two word augmentation performs best for Att-aug, except T5-base where three word augmentation performs best. Keyphrases up to 2-grams long perform best for P2T, except T5-large where 3-grams perform best. All models perform best with keyphrases up to 5-grams long for MI. These are the results reported here, and graphs displaying other hyperparameter results on test$_{CG}$ are in Appendix D. Table 14 contains qualitative examples, and more can be found in Appendix §E.

6.1 Automatic Evaluation

We see from Tables 7 to 10 that SAPPHIRE methods outperform the baselines on most/all metrics across the models on test$_{CG}$. The only exception is MI, which performs worse other than coverage.

For BART-base, Kw-aug, Att-aug, and P2T all outperform the baseline across the metrics. For BART-large, Att-aug and P2T outperform the baseline heaviest, with noticeable increases to all metrics. For T5-base, all methods outperform the baseline, with Kw-aug performing best. Att-aug performs best for T5-large, and SAPPHIRE appears relatively less effective for T5-large. T5-large is the strongest baseline, and hence further improving its performance is possibly more difficult.

MI performs worse across most metrics except coverage, likely as MI almost always keeps inputs intact in their exact form. This is however possibly one reason for its low performance; it is less flexible. Further, as discussed in §4.2.2, MI is highly dependent on the input order. See §6.3 for more.

Table 11 contains statistical significance p-values from Pitman's permutation tests (Pitman, 1937) for what we adjudged to be the best performing method(s) per model compared to corresponding baselines on test$_{CG}$. Most metrics across the methods are significant compared to the baselines.

From Table 12, we see that SAPPHIRE models outperform the corresponding baselines reported in Lin et al. (2020) on test$_{O}$. T5-large-KW and P2T outperform EKI-BART (Fan et al., 2020) and KG-BART (Liu et al., 2021) on both SPICE and BLEU-4, which are two SOTA published CommonGen models that use external knowledge from corpora and KGs. As SPICE is used to rank the CommonGen leaderboard[5], T5-large-KW and P2T would place highly. SAPPHIRE models do lag behind the SOTA published RE-T5 (Wang et al., 2021), showing potential for further improvement. Further, the BART-large SAPPHIRE models perform worse than EKI-BART and KG-BART, but not by a substantial margin. We emphasize again that SAPPHIRE simply uses the data itself and the baseline generations, rather than external knowledge. Hence, SAPPHIRE's performance gains over the baselines and certain SAPPHIRE models matching or outperforming SOTA models that leverage external information is quite impressive.

6.2 Human Evaluation

Table 13 shows human evaluation results on test$_{CG}$ for human references and methods (excluding MI) using BART-large and T5-base. SAPPHIRE generally outperforms the baselines. BART-large-P2T performs noticeably higher on both fluency and commonsense. For T5-base, all three methods outperform the baseline across both metrics. Compared to humans, our best methods have comparable fluency, but still lag noticeably on commonsense, demonstrating that human-level generative commonsense reasoning is indeed challenging.

6.3 Qualitative Analysis

We see from Table 14 that many baseline outputs contain issues found in §3.2, e.g. incomplete or illogical sentences. Human references are fluent, logical, and sometimes more creative (e.g. example 5), which all methods still lack in comparison.

[5] https://inklab.usc.edu/CommonGen/leaderboard.html

	BART-base				
Metrics\Methods	Baseline	Kw-aug	Att-aug	P2T	BART-base-MI
ROUGE-1	43.96±0.03	**45.01**±0.00	44.99±0.10	44.87±0.42	44.83
ROUGE-2	17.31±0.02	**18.33**±0.06	18.18±0.04	18.04±0.13	17.44
ROUGE-L	36.65±0.00	37.28±0.24	**37.76**±0.12	37.28±0.11	34.47
BLEU-1	**73.20**±0.28	73.00±0.85	73.00±0.14	73.15±1.06	69.90
BLEU-2	54.50±0.14	55.35±0.49	**55.70**±0.28	55.65±0.35	49.00
BLEU-3	40.40±0.14	41.35±0.21	41.40±0.28	**41.85**±0.35	34.70
BLEU-4	30.10±0.14	31.10±0.14	30.95±0.07	**31.75**±0.35	24.70
METEOR	30.35±0.35	30.50±0.28	30.70±0.14	**31.05**±0.49	29.70
CIDEr	15.56±0.10	**16.18**±0.12	15.68±0.00	16.14±0.33	14.43
SPICE	30.05±0.07	30.45±0.07	30.65±0.35	**30.95**±0.21	28.40
BERTScore	59.19±0.32	59.32±0.25	**59.72**±0.03	59.54±0.05	53.73
Coverage	90.43±0.17	91.44±0.95	91.23±0.21	91.47±2.93	**96.23**

Table 7: Automatic evaluation results (with standard deviations) for BART-base on test$_{CG}$, averaged over two seeds for trained models. Bold corresponds to best performance on that metric.

	BART-large				
Metrics\Methods	Baseline	Kw-aug	Att-aug	P2T	BART-large-MI
ROUGE-1	45.67±0.25	46.71±0.05	**46.78**±0.14	46.26±0.29	41.69
ROUGE-2	18.77±0.04	19.64±0.05	**19.92**±0.19	19.37±0.17	15.40
ROUGE-L	37.83±0.29	38.38±0.01	**38.53**±0.03	38.22±0.16	33.32
BLEU-1	74.45±0.21	76.20±0.99	76.55±0.92	**77.10**±0.85	63.90
BLEU-2	56.25±0.78	58.60±0.14	**59.60**±0.00	58.95±0.64	42.40
BLEU-3	42.15±0.49	44.00±0.00	**45.20**±0.42	44.70±0.14	29.20
BLEU-4	32.10±0.42	33.40±0.28	**34.50**±0.42	34.25±0.21	20.50
METEOR	31.70±0.14	32.60±0.57	32.65±0.49	**33.00**±0.14	28.30
CIDEr	16.42±0.09	17.37±0.08	17.49±0.49	**17.50**±0.02	12.32
SPICE	31.85±0.21	33.15±0.49	33.30±0.28	**33.60**±0.00	26.10
BERTScore	59.95±0.29	60.83±0.29	60.87±0.45	**61.30**±0.66	48.56
Coverage	94.49±0.53	96.74±1.20	96.02±1.17	**97.02**±0.15	95.33

Table 8: Automatic evaluation results (with standard deviations) for BART-large on test$_{CG}$, averaged over two seeds for trained models. Bold corresponds to best performance on that metric.

	T5-base				
Metrics\Methods	Baseline	Kw-aug	Att-aug	P2T	BART-base-MI
ROUGE-1	44.63±0.13	46.42±0.01	**46.75**±0.11	45.73±0.27	44.92
ROUGE-2	18.40±0.14	**19.36**±0.24	19.20±0.17	18.51±0.11	17.98
ROUGE-L	37.60±0.16	**38.68**±0.08	38.51±0.21	38.07±0.10	34.88
BLEU-1	73.60±0.85	**76.25**±0.35	76.00±0.28	75.65±1.20	70.20
BLEU-2	57.00±0.71	**59.55**±0.64	58.75±0.35	58.15±0.64	50.50
BLEU-3	42.75±0.49	**45.10**±0.85	44.00±0.28	43.45±0.07	36.20
BLEU-4	32.70±0.42	**34.45**±0.92	33.30±0.28	33.10±0.28	26.10
METEOR	31.05±0.49	31.85±0.07	31.90±0.14	**32.05**±0.35	30.20
CIDEr	16.26±0.25	**17.37**±0.04	17.04±0.21	16.84±0.11	14.83
SPICE	31.95±0.07	32.75±0.21	32.85±0.21	**33.20**±0.14	29.70
BERTScore	61.40±0.34	**61.88**±0.06	61.28±0.10	61.46±0.01	55.04
Coverage	90.96±1.77	94.92±0.45	96.00±0.03	94.78±0.83	**96.03**

Table 9: Automatic evaluation results (with standard deviations) for T5-base on test$_{CG}$, averaged over two seeds for trained models. Bold corresponds to best performance on that metric.

	T5-large				
Metrics\Methods	Baseline	Kw-aug	Att-aug	P2T	BART-large-MI
ROUGE-1	46.26±0.17	**47.47**±0.16	47.40±0.12	46.72±0.26	42.78
ROUGE-2	19.62±0.17	20.02±0.07	**20.19**±0.01	19.76±0.22	16.61
ROUGE-L	39.21±0.22	39.84±0.12	**39.97**±0.06	39.19±0.09	34.52
BLEU-1	77.45±0.21	78.70±0.28	**78.95**±0.07	77.90±0.57	66.80
BLEU-2	60.75±0.21	62.10±0.14	**62.35**±0.07	61.00±0.42	45.90
BLEU-3	46.60±0.14	47.65±0.21	**47.95**±0.21	46.75±0.49	32.70
BLEU-4	36.30±0.00	36.80±0.28	**37.35**±0.49	36.10±0.42	23.90
METEOR	33.30±0.14	33.55±0.07	**33.70**±0.00	33.35±0.21	29.10
CIDEr	17.90±0.15	18.40±0.18	**18.43**±0.10	17.89±0.08	13.34
SPICE	34.25±0.07	**34.50**±0.28	33.70±0.14	34.00±0.28	28.00
BERTScore	62.65±0.07	**62.91**±0.15	62.78±0.21	62.46±0.11	50.57
Coverage	94.23±0.21	95.92±0.02	**96.08**±0.09	95.44±0.58	96.03

Table 10: Automatic evaluation results (with standard deviations) for T5-large on test$_{CG}$, averaged over two seeds for trained models. Bold corresponds to best performance on that metric.

	BART-base	BART-large		T5-base	T5-large
p-values	P2T	Att-aug	P2T	Kw-aug	Att-aug
ROUGE-1	1.58E-05	1.58E-05	7.58E-04	1.58E-05	1.58E-05
ROUGE-2	6.32E-05	1.58E-05	2.18E-03	1.58E-05	2.20E-03
ROUGE-L	6.32E-05	8.53E-04	2.78E-02	1.58E-05	1.58E-05
BLEU-1	**3.63E-01**	1.39E-04	6.94E-05	6.94E-05	1.11E-03
BLEU-2	1.11E-03	6.94E-05	6.94E-05	6.94E-05	5.69E-03
BLEU-3	3.26E-02	1.04E-03	9.03E-04	4.17E-04	3.40E-02
BLEU-4	**5.68E-02**	**1.57E-01**	8.40E-03	1.83E-02	**2.66E-01**
METEOR	1.57E-02	9.03E-04	6.94E-05	2.08E-04	**7.27E-01**
CIDEr	6.25E-04	2.08E-04	6.94E-05	6.94E-05	5.07E-03
SPICE	1.53E-03	6.25E-04	6.94E-05	1.43E-02	**9.16E-01**
BERTScore	3.33E-03	1.58E-05	1.58E-05	1.58E-05	**1.42E-01**
Coverage	3.16E-05	1.58E-05	1.58E-05	1.58E-05	1.58E-05

Table 11: Statistical significance p-values (from Pitman's permutation tests) for the best performing method(s) per model compared to the corresponding baselines. Insignificant p-values (using $\alpha = 0.05$ or 5E-02) are bolded.

For example 1, the baseline generation *"hands sitting on a chair"* misses the concept *"toy"*, whereas our methods do not. Kw-aug and Att-aug output complete and logical sentences. For example 2, the baseline generation of *"a camel rides a camel"* is illogical. Our methods output more logical and specific sentences. For example 3, our methods generate more complete and coherent sentences than the baseline, which lacks a subject (does not mention *who* is *"walking"*). For example 4, the baseline generation *"bus sits on the tracks"* is illogical as buses park on roads. Our methods do not suffer from this and output more reasonable text. For example 5, the baseline generation *"A lady sits in a sunglass."* is completely illogical. Kw-aug, Att-aug, and P2T all output logical text. For example 6, the baseline output *"Someone stands in front of someone holding a hand"* is generic and bland. Kw-aug, Att-aug, and P2T all output more specific and detailed text rather than simply referring to *"someone"*. Overall, SAPPHIRE generates text that is more complete, fluent, logical, and with greater coverage, addressing many baseline issues (§3.2).

However, SAPPHIRE methods are imperfect. P2T relies heavily on the original generation. For example 1, the baseline output *"hands sitting on a chair"* is extracted as a keyphrase, and used in the P2T output *"hands sitting on a chair with toys"*. While coverage improves, the text is still illogical. For example 2, P2T still misses the *"walk"* concept. While the Att-aug output of *"A man is riding camel as he walks through the desert."* is more logical than the baseline's, it is still not entirely logical as the man cannot ride the camel and walk at the same time. MI outputs logical and fluent text for examples 2 and 3. For the other examples, the generated texts are illogical, not fluent, or incomplete.

This is likely due to input permutation having a strong effect on output quality. For example, *"wave"* before *"falls off a surf board"* leads to an illogical output *"A wave falls off a surf board."*, where the reverse order results in a more logical output *"A man falls off a surf board and hits a wave."* As discussed in §4.2.2, our method of selecting best permutations is likely imperfect. Further, BART-MI usually does not inflect inputs and retains them in exact form, unlike the baselines and other methods which learn to inflect words (e.g. singular to plural). We believe BART-MI has potential if these weaknesses can be addressed.

7 Related Work

Constrained Text Generation: There has been more work on constrained text generation. Miao et al. (2019) use Metropolis-Hastings sampling to determine token-level edits at each step of generation. Feng et al. (2019) introduce Semantic Text Exchange to adjust the semantics of a text given a *replacement entity*. Gangal et al. (2021a) propose narrative reordering (NAREOR) to rewrite stories in different narrative orders while preserving plot.

Data-to-text NLG: A wide range of data-to-text NLG benchmarks have been proposed, e.g. for generating weather reports (Liang et al., 2009), game commentary (Jhamtani et al., 2018), and recipes (Kiddon et al., 2016). E2E-NLG (Dušek et al., 2018) and WebNLG (Gardent et al., 2017) are two benchmarks that involve generating text from meaning representation (MR) and triple sequences. Montella et al. (2020) use target Wiki sentences with parsed OpenIE triples as weak supervision for WebNLG. Tandon et al. (2018) permute input MRs to augment examples for E2E-NLG.

Commonsense Reasoning and Incorporation: Talmor et al. (2020) show that not all pretrained LMs can reason through commonsense tasks. Other works investigate commonsense injection into models; one popular way is through knowledge graphs (KGs). One large commonsense KG is COMET, which trains on KG edges to learn connections between words and phrases. COSMIC (Ghosal et al., 2020) uses COMET to inject commonsense. EKI-BART (Fan et al., 2020) and KG-BART (Liu et al., 2021) show that external knowledge (from corpora and KGs) can improve performance on CommonGen. Distinctly, SAPPHIRE obviates reliance on external knowledge.

Models\Metrics	ROUGE-2/L		BLEU-3/4		METEOR	CIDEr	SPICE	Coverage
T5-base (reported baseline)	14.63	34.56	28.76	18.54	23.94	9.40	19.87	76.67
BART-large (reported baseline)	22.02	41.78	39.52	29.01	31.83	13.98	28.00	97.35
T5-large (reported baseline)	21.74	42.75	43.01	31.96	31.12	15.13	28.86	95.29
EKI-BART (Fan et al., 2020)	-	-	-	35.945	-	16.999	29.583	-
KG-BART (Liu et al., 2021)	-	-	-	33.867	-	16.927	29.634	-
RE-T5 (Wang et al., 2021)	-	-	-	**40.863**	-	**17.663**	**31.079**	-
BART-base-P2T	20.83	42.91	40.74	29.918	30.61	14.670	26.960	92.84
T5-base-P2T	22.38	44.59	44.97	33.577	31.95	16.152	29.104	95.55
BART-large-KW	22.25	43.38	43.87	32.956	32.26	16.065	28.335	96.16
BART-large-Att	22.22	43.80	44.61	33.405	32.03	16.036	28.452	96.43
BART-large-P2T	22.65	43.84	44.78	33.961	32.18	16.174	28.462	96.20
T5-large-KW	23.79	45.73	48.06	37.023	32.85	16.987	29.659	95.32
T5-large-Att	23.94	45.87	47.99	36.947	32.79	16.943	29.607	95.43
T5-large-P2T	23.89	45.77	48.08	37.119	32.94	16.901	29.751	94.82

Table 12: Automatic evaluation results of select SAPPHIRE models on test$_O$ (evaluated by the CommonGen authors). For BART-base and T5-base, we report the best SAPPHIRE model on test$_O$ (P2T), and all three models for BART-large and T5-large. We compare to Lin et al. (2020)'s reported baseline numbers, noting that they did not report BART-base, and published models on their leaderboard[5] that outperform the baselines at the time of writing. Bold corresponds to best performance (for BLEU-4, CIDEr, and SPICE, since their leaderboard only reports these three), and underline corresponds to second best performance.

Model	Method	Fluency	Commonsense
BART-large	Baseline	3.92	4.06
	Kw-aug	4.13	3.92
	Att-aug	4.10	4.06
	P2T	**4.17**	**4.13**
T5-base	Baseline	4.02	3.83
	Kw-aug	4.04	4.04
	Att-aug	**4.13**	3.98
	P2T	4.02	**4.08**
Human		4.14	4.32

Table 13: Avg. human eval results on test$_{CG}$, rated on 1-5 scales. Bold corresponds to best performance for that model.

8 Conclusion and Future Work

In conclusion, we motivated and proposed several improvements for concept-to-text generation which we call SAPPHIRE: Set Augmentation and Post-hoc PHrase Infilling and REcombination. We demonstrated their effectiveness on CommonGen through experiments on BART and T5. Extensive evaluation showed that SAPPHIRE improves model performance, addresses many issues of the baselines, and has potential for further exploration.

Potential future work includes improving mask infilling performance, and trying combinations of SAPPHIRE techniques as they could be complementary. Better exploiting regularities of CommonGen-like tasks, e.g. invariance to input order, presents another avenue. SAPPHIRE methods can also be investigated for other data-to-text NLG tasks, e.g. WebNLG, and explored for applications such as improving the commonsense reasoning of personalized dialog agents (Li et al., 2020), data augmentation for NLG (Feng et al., 2020, 2021), and constructing pseudo-references for long-context NLG (Gangal et al., 2021b).

Method	Text
Concept Set	{sit, chair, toy, hand} (example 1)
BART-base-BL	hands sitting on a chair
BART-base-KW	A boy sits on a chair with a toy in his hand.
BART-base-Att	A child sits on a chair with a toy in his hand.
BART-base-P2T	hands sitting on a chair with toys
BART-base-MI	Children's hands sit on a chair with a toy.
Human	A baby sits on a chair with a toy in one of its hands.
Concept Set	{camel, desert, ride, walk} (example 2)
BART-base-BL	a camel rides a camel in the desert
BART-base-KW	A camel rides down a walkway in the desert.
BART-base-Att	A man is riding camel as he walks through the desert.
BART-base-P2T	A camel rides down a trail in the desert.
BART-base-MI	In the desert, a man rides a camel for a walk.
Human	A loud group of people walk around the desert and ride camels.
Concept Set	{jacket, wear, snow, walk} (example 3)
BART-large-BL	walking in the snow wearing a furry jacket
BART-large-KW	A man wearing a jacket is walking in the snow.
BART-large-Att	A man in a blue jacket is walking in the snow.
BART-large-P2T	A man is wearing a furry jacket as he walks in the snow.
BART-large-MI	A walk in the snow wearing a furry jacket
Human	A man wears a jacket and walks in the snow.
Concept Set	{bench, bus, wait, sit} (example 4)
BART-large-BL	A bus sits on the tracks with people waiting on benches.
BART-large-KW	A bus sits next to a bench waiting for passengers.
BART-large-Att	A woman sits on a bench waiting for a bus.
BART-large-P2T	A bus sits at a stop waiting for passengers to get off the bench.
BART-large-MI	There are people waiting on benches outside bus stops to sit down. pic.twitter.
Human	The man sat on the bench waiting for the bus.
Concept Set	{sunglass, wear, lady, sit} (example 5)
T5-base-BL	A lady sits in a sunglass.
T5-base-KW	A lady sits next to a man wearing sunglasses.
T5-base-Att	A lady sits wearing sunglasses.
T5-base-P2T	A lady sits next to a man wearing sunglasses.
BART-base-MI	A young lady sits in a sunglass to wear.
Human	The lady wants to wear sunglasses, sit, relax, and enjoy her afternoon.
Concept Set	{hold, hand, stand, front} (example 6)
T5-large-BL	Someone stands in front of someone holding a hand.
T5-large-KW	Two men stand in front of each other holding hands.
T5-large-Att	A man stands in front of a woman holding a hand.
T5-large-P2T	A man standing in front of a man holding a hand.
BART-large-MI	Mr. Trump holding a hand to stand in front of
Human	A man stands and holds his hands out in front of him.

Table 14: Qualitative examples for test$_{CG}$. Color coded: baseline, Kw-aug, Att-aug, P2T, MI, and human reference.

Acknowledgments

We thank our anonymous reviewers, Graham Neubig, Ritam Dutt, Divyansh Kaushik, and Zhengbao Jiang for their comments and suggestions.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *European conference on computer vision*, pages 382–398. Springer.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, A. Jorge, C. Nunes, and A. Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *ECIR*.

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

Wenchao Du and Alan W Black. 2019. Boosting dialog response generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 38–43, Florence, Italy. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands. Association for Computational Linguistics.

Zhihao Fan, Yeyun Gong, Zhongyu Wei, Siyuan Wang, Yameng Huang, Jian Jiao, Xuanjing Huang, Nan Duan, and Ruofei Zhang. 2020. An enhanced knowledge injection model for commonsense generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2014–2025, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Steven Y. Feng, Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. 2020. GenAug: Data augmentation for finetuning text generators. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 29–42, Online. Association for Computational Linguistics.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online. Association for Computational Linguistics.

Steven Y. Feng, Aaron W. Li, and Jesse Hoey. 2019. Keep calm and switch on! preserving sentiment and fluency in semantic text exchange. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2701–2711.

Varun Gangal, Steven Y. Feng, Eduard Hovy, and Teruko Mitamura. 2021a. NAREOR: The narrative reordering problem. *arXiv preprint arXiv:2104.06669*.

Varun Gangal, Harsh Jhamtani, Eduard Hovy, and Taylor Berg-Kirkpatrick. 2021b. Improving automated evaluation of open domain dialog via diverse reference augmentation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4079–4090, Online. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

Deepanway Ghosal, Navonil Majumder, Alexander Gelbukh, Rada Mihalcea, and Soujanya Poria. 2020. COSMIC: COmmonSense knowledge for eMotion identification in conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2470–2481, Online. Association for Computational Linguistics.

Maarten Grootendorst. 2020. Keybert: Minimal keyword extraction with bert.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Learning to generate move-by-move commentary for chess games from large-scale social forum data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1661–1671.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 329–339.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Aaron W. Li, Veronica Jiang, Steven Y. Feng, Julia Sprague, Wei Zhou, and Jesse Hoey. 2020. ALOHA: Artificial learning of human attributes for dialogue agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8155–8163.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S. Yu. 2021. KG-BART: Knowledge graph-augmented bart for generative commonsense reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(7):6418–6425.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. CGMH: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842.

Sebastien Montella, Betty Fabre, Tanguy Urvoy, Johannes Heinecke, and Lina Rojas-Barahona. 2020. Denoising pre-training and data augmentation strategies for enhanced RDF verbalization with transformers. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 89–99, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Edwin JG Pitman. 1937. Significance tests which may be applied to samples from any populations. *Supplement to the Journal of the Royal Statistical Society*, 4(1):119–130.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. oLMpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758.

Shubhangi Tandon, TS Sharath, Shereen Oraby, Lena Reed, Stephanie Lukin, and Marilyn Walker. 2018. TNT-NLG, System 2: Data repetition and meaning representation manipulation to improve neural generation. *E2E NLG Challenge System Descriptions*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Han Wang, Yang Liu, Chenguang Zhu, Linjun Shou, Ming Gong, Yichong Xu, and Michael Zeng. 2021. Retrieval enhanced model for commonsense generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3056–3062, Online. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations 2020*.

Appendices

A Model Training and Generation Details

T5-large consists of 770M params, T5-base 220M params, BART-large 406M params, and BART-base 139M params. We train two seeded versions of each baseline model and SAPPHIRE model. For all models, we use beam search with a beam size of 5, decoder early stopping, a decoder length penalty of 0.6, encoder and decoder maximum lengths of 32, and a decoder minimum length of 1. For model training, we use a batch size of 128 for T5-base and BART-base, 32 for BART-large, and 16 for T5-large. For T5-base, T5-large, and BART-base, we use 400 warmup steps, and 500 for BART-large. We train all models up to a reasonable number of epochs (e.g. 10 or 20) and perform early stopping using our best judgment (e.g. if metrics have continually decreased for multiple epochs). The learning rates for SAPPHIRE models were determined by trying a range of values (e.g. from 1e-6 to 1e-4), and finding ones which led to good convergence behavior (e.g. validation metrics increase at a decently steady rate and reach max. after a reasonable number of epochs). For the best-performing models, learning rates are as follows (each set consists of {baseline,Kw-aug,Att-aug,P2T}): BART-base = {3e-05,2e-05,3e-05,1e-05}, BART-large = {3e-05,2e-05,2e-05,5e-06}, T5-base = {5e-05,5e-05,5e-05,1e-05}, T5-large = {2e-05,2e-05,2e-05,5e-06}.

Training was done using single RTX 2080 Ti and Titan Xp GPUs, and Google Colab instances which alternately used a single V100, P100, or Tesla T4 GPU. The vast majority of the training was done on a single V100 per model. T5-base models trained in approx. 1 hour, BART-base models in approx. 45 minutes, T5-large models in approx. 4 hours, and BART-large models in approx. 1.5-2 hours.

B Full Re-implementation versus Reported Model Numbers

See Table 16 for full comparison of our re-implemented CommonGen models compared to the original reported numbers in Lin et al. (2020).

C Human Evaluation Details

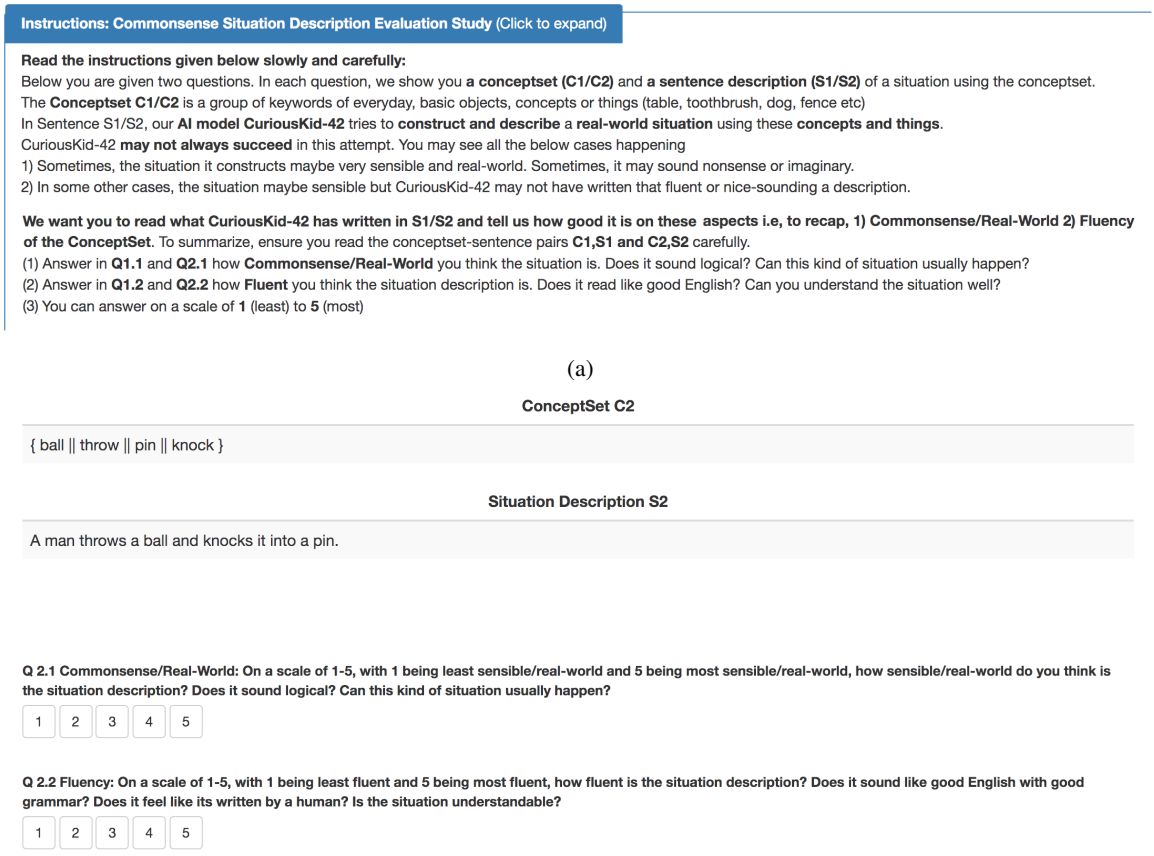
Human evaluation was done via paid crowdworkers on AMT, who were from Anglophone countries with lifetime approval rates $> 97\%$. Each example was evaluated by 2 annotators. The time given for each AMT task instance or HIT was 8 minutes. Sufficient time to read instructions, as calibrated by authors, was also considered in the maximum time limit for performing each HIT. Annotators were paid 98 cents per HIT. This rate (7.35$/hr) exceeds the minimum wage for the USA (7.2$/hr) and constitutes fair pay. We neither solicit, record, request, or predict any personal information pertaining to the AMT crowdworkers. Specific instructions and a question snippet can be seen in Figure 1.

Method	Text
Concept Set	{food, eat, hand, bird}
BART-base-BL	hands of a bird eating food
BART-base-KW	a bird eats food from a hand
BART-base-Att	hand of a bird eating food
BART-base-P2T	A bird is eating food with its hand.
BART-base-MI	The food is in the hands of a bird eating it.
Human	A small bird eats food from someone's hand.
Concept Set	{front, dance, routine, perform}
BART-base-BL	A woman performs a routine in front of a dance.
BART-base-KW	A man performs a routine in front of a group of people.
BART-base-Att	A man is performing a routine in front of a group of people.
BART-base-P2T	A woman performs a routine in front of a group of people.
BART-base-MI	In this dance, a man performs a routine in front of a mirror.
Human	The girl performed her dance routine in front of the audience.
Concept Set	{chase, ball, owner, dog, throw}
BART-base-BL	A dog is throwing a ball into a chase.
BART-base-KW	A dog is about to throw a ball to its owner.
BART-base-Att	A dog is trying to throw a ball at its owner.
BART-base-P2T	A dog is chasing the owner of a ball.
BART-base-MI	The dog was trained to throw balls and the dog would chase after the owner.
Human	The owner threw the ball for the dog to chase after.
Concept Set	{music, dance, room, listen}
BART-large-BL	A listening music and dancing in a dark room
BART-large-KW	A group of people dance and listen to music in a room.
BART-large-Att	A group of people are dancing and listening to music in a room.
BART-large-P2T	Two people are dancing and listening to music in a dark room.
BART-large-MI	Music and dancing in the dance floor.
Human	A boy danced around the room while listening to music.
Concept Set	{cheer, team, crowd, goal}
T5-base-BL	the crowd cheered after the goal.
T5-base-KW	the crowd cheered after the goal by football team
T5-base-Att	the crowd cheered after the goal by the team.
T5-base-P2T	the crowd cheered as the team scored their first goal.
BART-base-MI	The team and the crowd cheered after the goal.
Human	The crowd cheered when their team scored a goal.
Concept Set	{bag, put, apple, tree, pick}
T5-base-BL	A man puts a bag of apples on a tree.
T5-base-KW	A man puts a bag under a tree and picks an apple.
T5-base-Att	A man puts a bag under a tree and picks an apple.
T5-base-P2T	A man puts a bag of apples on a tree and picks them.
BART-base-MI	A man puts a bag of apple juice on a tree to pick it up
Human	I picked an apple from the tree and put it in my bag.
Concept Set	{circle, ball, throw, turn, hold}
T5-large-BL	Someone turns and throws a ball in a circle.
T5-large-KW	A man holds a ball and turns to throw it into a circle.
T5-large-Att	A man holds a ball in a circle and throws it.
T5-large-P2T	A man holds a ball, turns and throws it into a circle.
BART-large-MI	He turns and throws a ball into the circle to hold it.
Human	A girl holds the ball tightly, then turns to the left and throws the ball into the net which is in the shape of a circle.
Concept Set	{hair, sink, lay, wash}
T5-large-BL	A woman is washing her hair in a sink.
T5-large-KW	A woman lays down to wash her hair in a sink.
T5-large-Att	A man lays down to wash his hair in a sink.
T5-large-P2T	A woman is washing her hair in a sink.
BART-large-MI	A woman is washing her hair in the sink. She lay the sink down
Human	The woman laid back in the salon chair, letting the hairdresser wash her hair in the sink.
Concept Set	{wash, dry, towel, face}
T5-large-BL	A man is washing his face with a towel.
T5-large-KW	A man washes his face with a towel and then dries it.
T5-large-Att	A man is washing his face with a towel and drying it.
T5-large-P2T	A man is washing his face with a towel and drying it off.
BART-large-MI	A man is washing his face with a towel to dry it.
Human	The woman will wash the baby's face and dry it with a towel.

Table 15: Qualitative examples for test$_{CG}$. Color coded: baseline, Kw-aug, Att-aug, P2T, MI, and human reference.

Model\Metrics	ROUGE-2/L		BLEU-3/4		METEOR	CIDEr	SPICE	BERTScore	Cov
Reported BART-large	22.13	43.02	37.00	27.50	31.00	14.12	30.00	-	97.56
Reported T5-base	15.33	36.20	28.10	18.00	24.60	9.73	23.40	-	83.77
Reported T5-Large	21.98	44.41	40.80	30.60	31.00	15.84	31.80	-	97.04
Our BART-base	15.91	36.15	38.30	28.30	30.20	15.07	30.35	58.26	93.44
Our BART-large	17.27	37.32	**39.95**	**30.20**	**31.15**	**15.72**	**31.20**	58.58	95.03
Our T5-base	**17.27**	**37.69**	**41.15**	**31.00**	**31.10**	**16.37**	**32.05**	60.32	**94.44**
Our T5-large	17.90	38.31	**43.80**	**33.60**	**32.70**	**17.02**	**33.45**	61.39	96.26

Table 16: Performance of our re-implemented CommonGen models on dev_O compared to the original numbers reported in Lin et al. (2020). Note that for our models, results are averaged over two seeds, and that the original authors did not experiment with BART-base. Bold indicates where we match or exceed the reported metric.



(a)

(b)

Figure 1: Snapshots of human evaluation: a) instructions seen by annotator and b) an example with questions.

D Graphs Displaying Other Hyperparameter Results

Figures 2, 3, 4, and 5 contain graphs displaying other hyperparameter results for Kw-aug, Att-aug, P2T, and Mask Infilling (MI), respectively.

E Further Qualitative Examples

See Table 15 for further qualitative examples.

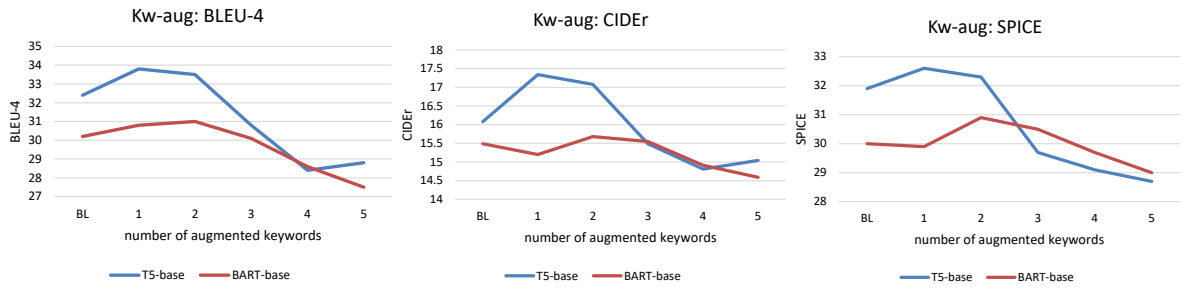


Figure 2: Kw-aug: graphs of BLEU-4, CIDEr, and SPICE results on test$_{CG}$ over different numbers of augmented keywords for BART-base and T5-base. These are only first seed results, and we only went above three augmented keywords for the base size models. BL refers to the baseline results with no augmented keywords.

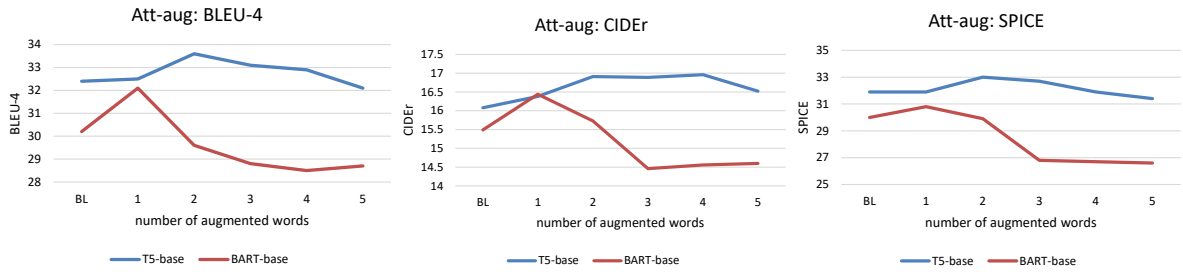


Figure 3: Att-aug: graphs of BLEU-4, CIDEr, and SPICE results on test$_{CG}$ over different numbers of augmented words for BART-base and T5-base. These are only first seed results, and we only went above three augmented words for the base size models. BL refers to the baseline results with no augmented words.

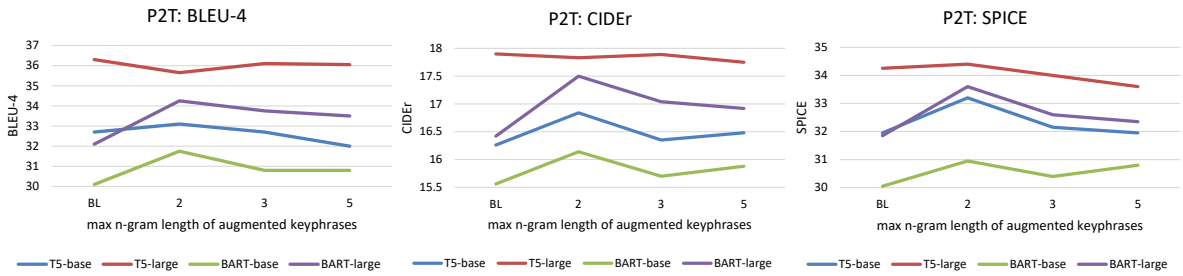


Figure 4: P2T: graphs of BLEU-4, CIDEr, and SPICE results on test$_{CG}$ over different max n-gram lengths of augmented keyphrases. These are results averaged over two seeds. BL refers to the baseline results with no augmented keyphrases.

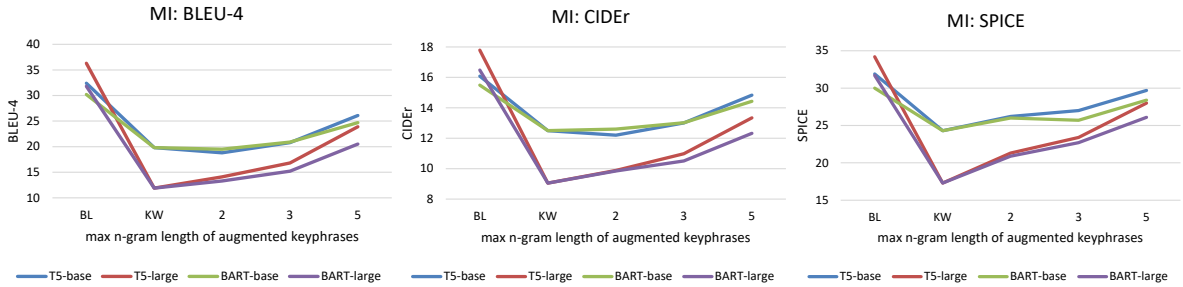


Figure 5: Mask Infilling (MI): graphs of BLEU-4, CIDEr, and SPICE results on test$_{CG}$ over different max n-gram lengths of augmented keyphrases. These are first seed results only. BL refers to the baseline results, and KW refers to mask infilling on the original keywords only (with no augmented keyphrases).

Contextualizing Variation in Text Style Transfer Datasets

Stephanie Schoch Wanyu Du Yangfeng Ji
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
{sns2gr,wd5jq,yangfeng}@virginia.edu

Abstract

Text style transfer involves rewriting the content of a source sentence in a target style. Despite there being a number of style tasks with available data, there has been limited systematic discussion of how text style datasets relate to each other. This understanding, however, is likely to have implications for selecting multiple data sources for model training. While it is prudent to consider inherent stylistic properties when determining these relationships, we also must consider how a style is realized in a particular dataset. In this paper, we conduct several empirical analyses of existing text style datasets. Based on our results, we propose a categorization of stylistic and dataset properties to consider when utilizing or comparing text style datasets.

1 Introduction

The general task of text style transfer involves rewriting source content in a target style. Currently, there are a number of text style transfer tasks with available data, such as formality (Rao and Tetreault, 2018), bias (Pryzant et al., 2020), sentiment (He and McAuley, 2016), humor or romance (Gan et al., 2017), offensiveness, (Nogueira dos Santos et al., 2018), authorship or time period (Xu et al., 2012), and personal attributes (Kang et al., 2019). While these specific tasks are often modeled in isolation, the general task definition remains consistent. As such, a natural question arises of what the relationship is between the stylistic variation of specific tasks.

Stylistic variation can arise from a number of factors such as communicative intent, topic, and speaker-receiver dynamics (Biber and Conrad, 2019), yet within the task of text style transfer, our view of a style is constrained to the context of each specific dataset. Therefore, understanding the tasks as well as the relationships between different tasks requires considering the stylistic properties and potential contextual and social factors (Hovy and Yang, 2021; Hovy, 2018) underpinning them, as well as the dataset characteristics (Bender and Friedman, 2018) and intersection of influences giving rise to the realization of style within a dataset.

From an application standpoint, considering these influences can provide a more comprehensive understanding of important textual features. There is a body of work already looking at how to identify generic features to increase target task performance (Li et al., 2019) or to compute similarity of textual features to select data for transfer learning (Ruder and Plank, 2017). In the context of text style transfer, these approaches first require understanding what features should be shared across tasks. For example, Zhang et al. (2020) leveraged the stylistic features shared between grammatical error correction data and formality to increase model performance on formality transfer datasets.

In addition to textual features such as stylistic properties, existing work also suggests that context of dataset creation should be taken into account when identifying compatible data or assessing possible out-of-distribution generalizability. For example, the similarity between how sentiment information is reflected in different domains affects adaptation performance (Li et al., 2019), and many models can achieve high performance on natural language inference tasks through task-limiting annotation artifacts (Gururangan et al., 2018; Poliak et al., 2018). In other words, factors such as data source and annotation method can create underlying textual features that can impact performance and limit generalizability. Thus, in combination, these existing works on leveraging inherent stylistic similarities (Zhang et al., 2020) or similar style-representations in different dataset domains (Li et al., 2019), as well as identifying task-limiting dataset properties (Gururangan et al., 2018; Poliak

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 226–239,
Aberdeen, Scotland, UK, 20-24 September 2021.

Dataset	Stylistic Task	Domain	Annotation	Size		
				Train	Dev	Test
Flickr	Romantic→Humorous	Image Captions	Manual	6k	500	500
Shakespeare	Shakespeare→Modern	Literature, SparkNotes	Automatic	18.4k	1.2k	1.5k
GYAFC-FR	Informal→Formal	Yahoo Answers (Online)	Manual	52k	2.8k	1.3k
GYAFC-EM	Informal→Formal	Yahoo Answers (Online)	Manual	52.6k	2.9k	1.4k
Biased-word	Subjective→Neutral	Wikipedia (Online)	Automatic	53.8k	700	1k
Fluency	Disfluent→Fluent	Telephone Conversations	Manual	173.7k	10.1k	7.9k

Table 1: An overview of the datasets used for exploratory analyses. Task describes the source-target direction used in our experiments and domain and annotation show general categorizations. Size provides statistics of the data splits, with standard, pre-existing data splits used when available.

et al., 2018) indicate that analysis of both stylistic properties and dataset characteristics, as well as the potential interdependencies between them, is warranted.

In this paper, we consider two primary categories of textual variation within the context of text style transfer: **stylistic characteristics** and **dataset characteristics**. We perform a series of empirical analyses to demonstrate the visible influence of both style and dataset characteristics on the performance of text style transfer models. Then, we present a categorization of style and dataset properties for consideration when utilizing or comparing style transfer datasets. Finally, we discuss the downstream applications for contextualizing variation in text style datasets, including multi-task learning, data selection, and generalizability. Our work and suggestions fall within the context of and align with recent work on incorporating social factors in natural language processing systems (Hovy and Yang, 2021) and characterizing datasets (Bender and Friedman, 2018).

2 Empirical Analyses

As an exploratory step, we question whether we can distinguish differences arising from style or dataset properties when comparing empirical results across datasets. We identify a set of aligned English datasets used for supervised text style transfer that exhibit differences ranging from style, annotation method, and domain. We further restrict our selection to datasets in which a single stylistic attribute is transferred between classes. Specifically, we look at **GYAFC-EM** & **GYAFC-FR** (Rao and Tetreault, 2018), **Shakespeare** (Xu et al., 2012), **Biased-word (Bias)** (Pryzant et al., 2020), **Fluency** (Wang et al., 2020; Godfrey et al., 1992), and **Flickr** (Gan et al., 2017). We provide dataset overviews in Table 1, with detailed dataset descriptions provided in Appendix A. We perform a preliminary qualitative analysis to get an initial impression of the data differences.

First Impression of Data: Of the six datasets, four were manually annotated and two were automatically annotated. For manually annotated datasets, GYAFC-EM and GYAFC-FR utilized crowdsourced rewrites, Flickr utilized crowdsourced sentences with only visual context shared between annotators, and Fluency utilized expert annotations of the target attribute. Both automatically annotated datasets (Bias, Shakespeare) were created through identification of existing data sources. While each style task is unique (other than two domains of GYAFC for formality), in terms of style we observe that Shakespeare has a significantly different temporal context than all other datasets, and Fluency involves a stylistic attribute that, ideally, the sentence pairs in all other datasets should possess.[1]

Beyond our qualitative observations, we perform an exploratory multi-task learning experiment, described in the following subsection.

2.1 Multi-Task Learning

As a toy experiment, we ask the question "*What would our results look like if we naively train on all style transfer tasks, with no considerations beyond the fact that the tasks share a general task defi-*

[1]Fluency is frequently a criteria used in text style transfer evaluation (Mir et al., 2019; Briakou et al., 2021; Prabhumoye et al., 2018).

nition?[2] We essentially ignore all considerations for style or dataset properties. Our expectation is that negative transfer will occur due to the lack of consideration for factors such as domain (Pan and Yang, 2009; Li et al., 2019)[3], but we are interested in whether all tasks share similar performance patterns or if performance on any tasks diverge from the overall set. If the latter, is there any intuitive explanation for the divergences?

We further expect that the degree of negative transfer will be impacted by the degree of difference of stylistic or data properties, relative to the full set of pre-training datasets. Specifically, we anticipate some level of alignment with our initial impression of the data: the alternate temporal context of Shakespeare may increase degree of negative transfer, yet the inherent stylistic connection with Fluency may lessen the degree of negative transfer.

Experimental Setup We utilize two experimental settings: GPT-2 directly fine-tuned on each dataset, and GPT-2 with multi-task pre-training on all datasets followed by fine-tuning on each target dataset. For both settings, we initialize GPT-2 with the pre-trained parameters from Radford et al. (2019). For our multi-task experimental setup, we follow prior works (Liu et al., 2015, 2019; Raffel et al., 2020) to perform multi-task learning for the baseline GPT-2 model (Wang et al., 2019): we initialize GPT-2 with the pre-trained parameters from Radford et al. (2019), then we jointly pre-train on all style tasks in a supervised manner and fine-tune on each individual style transfer task. [4]

For multi-task learning, we construct our pre-training dataset by randomly shuffling the training examples from all datasets. During pre-training, each training example from each individual task is seen at least once per epoch. All of the training examples in the largest dataset are seen exactly once per epoch, while all training examples for the smallest dataset are seen multiple times per epoch (proportional to the ratio between the training set size of the largest-scale task and the smallest-scale task). For the fine-tuning step, we leverage the multi-task pre-trained model and further fine-tune on each individual supervised task, saving the model with the lowest validation set loss as our final model for evaluation.

Dataset	Task	BLEU-og	BLEU-mt	%og
Shakespeare	shake2mod	24.47	11.33	0.463
Fluency	dis2fl	96.59	96.69	1.001
Flickr	rom2fun	8.14	7.18	0.882
GYAFC-EM	inf2fr	69.96	65.16	0.931
GYAFC-FR	inf2fr	75.16	74.72	0.994
Biased	subj2neut	93.73	93.41	0.996

Table 2: Experiments conducted using GPT-2, where BLEU-og represents directly fine-tuning the original GPT-2 on the target task, BLEU-mt represents multi-task pre-training using all datasets and fine-tuning on the target task, and %og represents the relative performance of multi-task pre-training in comparison to the performance of the original GPT-2 (computed by dividing BLEU-mt by BLEU-og).

Results We report BLEU (Papineni et al., 2002) in Table 2 as a measure of content preservation.[5] We compare the performance between directly fine-tuning the original GPT-2 on the target task (BLEU-og) and firstly multi-task pretraining the original GPT-2 then fine-tuning it on the target task (BLEU-mt).

Negative transfer is identified as a performance drop in BLEU-mt, i.e. %og < 1.00. Since the style transfer datasets in use are diverse across domain and stylistic properties, we expect negative transfer to occur in the multi-task learning setting. However, we are specifically looking at the overall performance pattern as an initial step in determining what properties may underlie such differences, which should be accounted for in a taxonomy.

While most tasks perform within a 12% margin below the original GPT-2 performance, we observe two divergences: with multi-task learning, the Shakespeare-to-modern task performed at less than 50% of the original GPT-2 performance, and the disfluent-to-fluent task experienced a slight performance increase. Performance on Fluency exceeded our initial expectation that the degree of negative transfer would simply be lower compared to other datasets, but overall the divergences with Shakespeare and Fluency match our expectations based on our initial impression of the data style differences. Specifically, we attribute the performance drop on the Shakespeare dataset to limited suitability for combining the data sources likely due to the stylistic attribute pertaining to different temporal

[2]The general task definition is rewriting the source content of a text in a target style (see section 1)

[3]Negative transfer occurs when transferred knowledge negatively impacts target performance (Pan and Yang, 2009).

[4]GPT-2 models were each trained on a single NVIDIA GTX 1080 Ti GPU.

[5]We use the BLEU implementation from Koehn et al. (2007).

context, and we attribute the Fluency dataset performance increase to high suitability for combining the data sources likely due to its stylistic attribute pertaining to a textual criteria that is assumed to be inherent to the other data.

With regard to dataset differences, we note the potential impact of dataset size on performance: to maintain consistency of the model architecture, we utilize the same model configuration with GPT-2 across datasets and experimental settings. In the case of performance on the Flickr dataset (see Table 1), it is possible that such a model configuration may overfit on the dataset. However, this alone fails to account for our observations of performance pattern divergences.

Beyond overall pattern, we observe an unexpectedly wide range of BLEU scores across datasets, which we expect could be attributable to differences in either dataset creation or style. There may be stylistic differences in how style information is encoded that impact content preservation. For example, some styles may have more words that encode both style and content information which may increase the difficulty of content retention (Cao et al., 2020), yet other styles may be characterized by stylistic attributes encoded in only a few key words or phrases (Fu et al., 2019). However, these differences may also be attributable to dataset creation. We expect that if the attribute-encoding words are constrained to a few words or phrases as a property of a style itself, then a dataset's style classes should be highly distinguishable using lexical features; in other words, the decision boundary when classifying styles should stay at the lexical level (Fu et al., 2019).

To test these hypotheses and help explain the range of BLEU scores, we perform two complementary experiments. First, we compute sentence similarity metrics averaged over each dataset to 1) identify if there is a relationship between BLEU scores and baseline sentence pair similarities, and 2) identify datasets with high similarity across class boundaries that constrain stylistic attributes to a few words or phrases. Second, we perform classification and ablation studies using a set of linguistic features defined on each dataset. For datasets with high sentence similarities, if a style can be well-represented by a few style-encoding words or phrases, then we expect high classification performance using only lexical features. Conversely, if a style cannot be isolated to a few words and phrases, we expect low classification performance using lexical features alone, in which case a high sentence similarity is likely attributable to dataset properties rather than inherent style properties.

Dataset	JS ↑	LD ↓	LD-norm ↓	F1-Score ↑
Shakespeare	0.0845	14.79	0.9029	0.0583
Fluency	0.9941	0.366	0.0271	0.9751
Flickr	0.2257	11.92	0.7728	0.3623
GYAFC-EM	0.4471	7.924	0.5616	0.4207
GYAFC-FR	0.4565	7.723	0.5375	0.4500
Biased	0.9137	2.529	0.0763	0.9689

Table 3: Jaccard Similarity (JS), Levenshtein Distance (LD), normalized Levenshtein Distance (LD-norm), and F1-Score. Sentence similarity measures quantify the distance between target and source for the training sets with arrows indicating direction for more similar sentences.

2.2 Similarity Metrics

We calculate token-based Jaccard Similarity, token-based Levenshtein distance, and F_1-score between the source and target training sets. We also report Levenshtein distance normalized by sentence length, $LD_{norm}(\boldsymbol{s}, \boldsymbol{t}) = \left(\frac{LD(\boldsymbol{s},\boldsymbol{t})}{\max |\boldsymbol{s}|,|\boldsymbol{t}|}\right)$ where $LD(\boldsymbol{s}, \boldsymbol{t})$ is the Levenshtein distance, $\boldsymbol{s}, \boldsymbol{t}$ refer to sentences in a sentence pair, and $|\cdot|$ refers to the number of tokens in a sentence. Scores are reported in Table 3.[6]

We see some relationships between similarities in Table 3 and GPT-2 performances in Table 2 in that the datasets with the lowest BLEU scores (Shakespeare and Flickr) have the lowest baseline similarities, and the datasets with the highest BLEU scores (Fluency and Bias) have the highest baseline similarities. We therefore can identify the Fluency and Bias datasets as being of particular relevance for the lingusitic features analysis. Specifically, our hypothesis is that if the Bias and Fluency styles can truly be isolated to few words as the sentence similarities would suggest, then the classification performance should be high using only lexical features. In contrast, if the dataset properties influence variation through constrained stylistic representation, then we expect low classification accuracy using lexical features.

[6]We do not distinguish between source and target direction as these metrics are symmetric in our setting (see Appendix B).

Group	Features
Lexical Complexity	Average word length, average syllable count (with & without stopwords)
Readability	# complex words (≥ 3 syllables)*, Flesch Reading Ease Score, Flesch-Kincaid Grade Level
Lexical Diversity	Unique unigrams & bigrams, with punctuation removed*
POS tags	Universal POS tag distribution*, Penn Treebank POS tag distribution*
Sentence length	Sentence length (words & total tokens)
Phrases	# noun phrases*, # verb phrases*, average length of noun phrases*, average length of verb phrases*, # dependent clauses*, average length of dependent clauses*
Subjectivity	# 1st, 2nd, & 3rd person pronouns*, Subjectivity & Sentiment polarity according to TextBlob sentiment module
Bag-of-Words	Bag-of-words feature representation

Table 4: Linguistic feature groups: lexical (top), syntactic (gray in middle), and other (bottom). Features features denoted with an asterisk (*) are normalized by sentence length.

2.3 Linguistic Features Analysis

We define linguistic features to refer to properties characterizing textual variation primarily at the lexical or syntactic level, where the "other" category in Table 4 indicates features that may capture slight semantic variation (subjectivity) or reflect overall lexical tendencies (bag-of-words). Features are adopted from prior works (Pavlick and Tetreault, 2016; Abu-Jbara et al., 2011; Roemmele et al., 2017) and listed in Table 4, with further description in Appendix C.

We train logistic regression classifiers with $\ell 1$-regularization and feature scaling on the full feature set for each text style dataset. Next, we train and subsequently test classifiers with all features ablated except the specified subset, and identify important features as those with minimal relative performance drop compared to full-feature classification accuracy. Results are shown in Table 5. We further quantify the magnitude of variation by computing the Jensen-Shannon (JS) divergence for each feature, and indicate the cells corresponding to features with divergences ≥ 0.075 in Table 5 in bold.[7]

Datasets with the lowest BLEU scores (Flickr and Shakespeare) have more distributed salient class features across linguistic levels, further reflected in a higher number of features with large divergence magnitudes (≥ 0.075). For the high BLEU and sentence similarity datasets of interest (Bias, Fluency), the inverse of this is true. For Bias and Fluency we see consistently low classification

[7]Table 6 in Appendix D shows a JS-divergence heatmap.

	Flick	Shake	GY-FR	GY-EM	Bias	Flu.
FF	75.6	76.1	80.7	80.9	63.5	55.3
LexC	**51.7**	62.2	**65.6**	64.4	52.6	50.7
Read	**55.7**	52.1	**62.1**	63.3	52.0	51.0
LexD	52.4	49.6	51.2	52.0	**50.4**	**54.4**
UPOS	**59.4**	59.3	62.3	**60.8**	54.4	51.6
XPOS	62.3	59.7	65.1	66.1	55.0	51.7
SenL	**51.8**	**56.7**	56.2	51.7	50.3	51.0
Phr	**54.2**	58.2	53.6	53.4	52.9	51.8
Sub	**60.5**	**60.4**	51.7	52.9	57.0	50.4
BoW	74.2	72.4	71.5	71.7	62.2	50.3

Table 5: Classification accuracy using linguistic feature groupings described in Table 4, with Full Features (FF) indicating the entire suite of features. Classification accuracy for features with Jensen-Shannon divergences ≥ 0.075 are in bold.

performance across ablations, including the lexical feature ablations. These results support our hypotheses and further suggest that neither stylistic differences nor dataset characteristics alone can be used to relate text style datasets. Rather, both influences as well as their interactions require consideration.

In the following section, we propose a taxonomy of style and dataset property categories that can contribute to variation in text style transfer datasets. Additionally, we note that when introducing these properties, we view style *as the targeted stylistic property within the context of a text style dataset*.

3 Variation From Style and Data Properties

Our empirical analyses demonstrate the visible influence of both style and dataset properties on how a style is represented in a given dataset. In addition to brief mentions of influences of dataset creation in section 1, we can identify an intuitive reason for these dual influences. While linguistic approaches exist to analyze textual variation (Halliday and Matthiessen, 2013; Holmes and Wilson, 2017; Biber, 2012), we suggest that the processes of linguistic-based stylistic analysis and text style transfer typically occur in inverse directions: linguistic analysis may work from human-written text and then analyze stylistic variation, whereas text style transfer may work from pre-existing ideas of targeted stylistic variation and then create datasets of human-written text that meet stylistic expectations. In other words, to create a text style transfer dataset or train a text style transfer model, the researcher should have a notion of the desired style against which to judge the resulting artifact. Intuitively, this process can lead to process-attributable

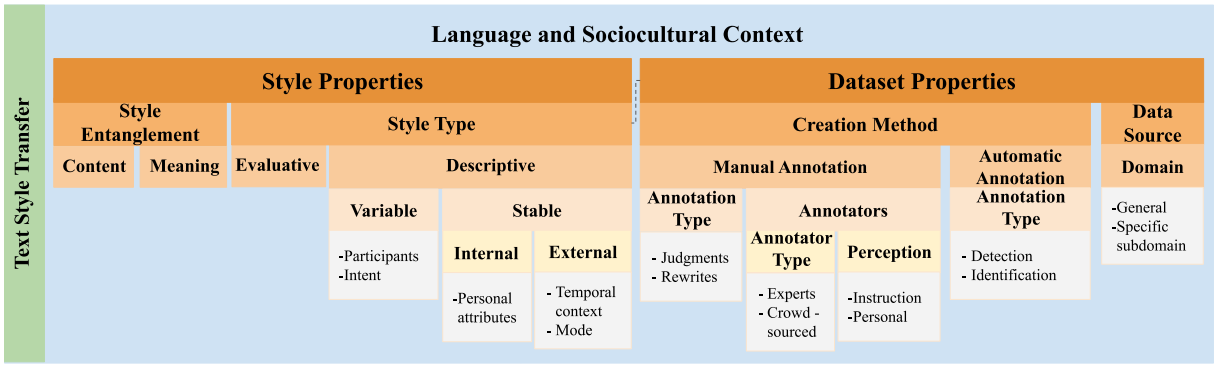


Figure 1: Framework overview visualizing style and dataset properties discussed throughout section 3. Boxes with bullet points indicate example considerations within each category. We contextualize both style and dataset properties within language and sociocultural context as all language is implicitly reflective of these influences (Hovy and Yang, 2021).

variation secondary to and alongside the intended stylistic variation.

Based on our results and observations, we consider ***stylistic properties*** as properties influencing textual variation that are inherent to a particular style and ***dataset properties*** as factors influencing textual variation due to how a particular dataset was created. We detail style and dataset properties in the following subsections and visualize the major distinctions in Figure 1.

3.1 Stylistic Properties

We group stylistic properties under two broad categories: *style entanglement* and *style type*.

3.1.1 Style Entanglement

Although some recent approaches to style transfer model style and content words separately (Li et al., 2018), or try to disentangle style and content representations (John et al., 2019; Kazemi et al., 2019), this approach may be less effective when used to transfer styles in which a higher ratio of words embed both style and content information. We can consider this ratio of dual-embedding a property inherent to the style. Specifically, we can consider how entangled the style and the content or semantic meaning is, where *content entanglement* refers to whether changes to the style result in additions or reductions in the total content details, and *meaning entanglement* refers to whether changes to the style can retain the content details but alter the semantic meaning. As an example of this distinction, sentiment transfer, which has been regarded previously as transfer between negative and positive style (Shen et al., 2017; Prabhumoye et al., 2018) alters semantic meaning while retaining most content, yet transferring between styles such as expert-to-layman can retain meaning but lead to content detail reductions due to the difficulty of preserving content from professional sentences (Cao et al., 2020).

3.1.2 Style Type

Style can refer to the *individuating* sense or *evaluative* sense of a text (Crystal and Davy, 1969). We refer to ***evaluative styles*** as styles distinguished by general properties that address overall textual quality corresponding with rules of usage and composition, effectiveness of expression (Strunk and White, 1999) or based on overall quality evaluation and judgments (Williams and Bizup, 2017). Stylistic variation occurs solely along evaluative lines, independent of situational context or language choice. From our empirical experiments, we can consider the Fluency dataset representative of a dataset in which the transferred stylistic attribute refers to an evaluative sense of style.

We consider ***descriptive styles*** as distinguished by stylistic properties that characterize textual variation through influences such as the underlying communicative intent, the situational or social factors influencing language choice, and the attributes of the producer of the text. We can further differentiate descriptive styles by the stability or variability of the targeted stylistic property.

Stability of Targeted Style Properties On one end of the spectrum variable stylistic properties (high variance, low stability) are characterized by dynamically shifting language to convey information a certain way, which may be reflective of factors such as the underlying intent in producing the text or the social dynamics of a situation. For exam-

ple, politeness can shift based on social dynamics such as social distance and relative power between participants (Brown et al., 1987) independently of the directness of communication, such as formality [8] in email (Peterson et al., 2011). From our empirical experiments, we consider Flickr, GYAFC, and Bias as reflective of variable targeted properties.

At the other end of the spectrum, more stable targeted stylistic properties (low variance, high stability) remain more consistent across social situations and arise from relatively stable internal or external context. These may reflect internal context such as the personal attributes of the producer of text (Kang et al., 2019), or external context such as the temporal context at time of text production or stylistic properties inherent to the mode of distribution. Example datasets include the PASTEL dataset (Kang et al., 2019) annotated for personal attributes such as gender and age group, and the Shakespeare dataset (Xu et al., 2012) which can be considered reflective of authorship (Xu, 2017) or temporal context.[9]

3.2 Dataset Properties

While in the previous section we discussed properties inherent to specific styles, in this section we discuss properties of datasets to which textual variation is attributable. We identify the broad categories of properties due to *creation method* and *data source*. In this context, creation method refers to the general method of creating sentence pairs (automatic or manual annotation, as well as any properties arising from utilizing a specific method, such as influences of annotator background or perceptions) and data source refers to characteristics (such as domain) from where the source data was collected. We provide more detailed discussion in the following subsections.

3.2.1 Creation Method

Generally speaking, datasets can be created via manual annotation, such as through judgments or rewrites, or via automatic annotation, such as through filtering data that has a target attribute (i.e., detection with a classifier). With particular attention on manual annotation, in addition to potential generalizability-limiting data properties arising from artifacts of the *annotation method* and *annotation type* ((Geva et al., 2019), also, see section 1), the *annotators* themselves can influence stylistic variation. For example, model performance has been improved by incorporating annotator identifiers as features (Geva et al., 2019) and by augmenting machine translation models with distinct translator styles identifiable in the training data (Wang et al., 2021). In the case of Wang et al. (2021), using annotator styles resulted in BLEU score variations of up to $+4.5$ points.

Underlying these influences, annotator properties that may give rise to textual variation could include the background of the annotator such as experts or crowd-sourced workers, and the perception the annotators have of the style task. Similar to human evaluation of outputs, perception may arise due to personal understanding or the wording of instructions presented.[10].

Data Source - Domain: Differences in domain can be reflected in entirely different word meanings and contexts of use (Li et al., 2019), as well as different manners of encoding attribute information such as sentiment (Blitzer et al., 2007; Li et al., 2019). In addition to differences of a single style between domains, the domains themselves have different levels of stylistic diversity (Kang and Hovy, 2021). Further, while the properties characterizing a style may be inherent to how a style is realized *within* a domain, there is a distinction in how the style is reflected *between* domains that necessitates domain being considered as a dataset property influencing variation in text style datasets.

4 Interplay Between Style and Data Properties

Bender and Friedman (2018) proposed data statements for documenting dataset contextual factors such as language variety, speaker demographics, annotator demographics, speech situation, and text characteristics (e.g. genre, topic). The style and dataset properties we discuss as potentially contributing to variation in text style transfer datasets show some alignment with those proposed for data statements as such factors contribute to linguistic variation in a general sense. However, our categorization specifically operates within the context

[8] Formality is closely related to politeness (Kang and Hovy, 2021)

[9] Regarding distribution mode, Abu-Jbara et al. (2011) suggested a set of linguistic features differentiating written and audio styles.

[10] Schoch et al. (2020) discuss potential influences of framing effects of questions or instructions on results in human evaluation of outputs, and we suggest similar effects could influence dataset properties resulting from annotation of inputs.

of text style transfer datasets for which there are unique considerations and important distinctions between sources of variation and downstream implications or applications.

In the previous subsections, we discussed style properties and dataset properties to which variation in text style transfer datasets can be attributed. In this section, we discuss the interdependence of style and data properties in text style transfer datasets in terms of context-dependence of and interactions between sources of variation.

Style and Data Property Interactions While we previously considered the potential impact of both style and dataset characteristics independently, these characteristics may have underlying interactions and influences on one another. Specifically, certain types of stylistic properties may be more or less amenable to certain dataset creation methods or sources, and vice versa.

With regard to the stability of stylistic properties, dataset properties such as annotation method may be indirectly influenced when transferring across relatively stable stylistic properties. For example, machine translation models have been found to exhibit stylistic bias through reflecting demographically-biased training data (Hovy et al., 2020). While this demonstrates that the demographics of annotators can serve as an important dataset characteristic, it also demonstrates the potential to transfer across relatively stable stylistic properties, such as personal attributes (Kang et al., 2019). However, as the stylistic properties are inherent to the annotator, there may be constraints on dataset creation through manual data annotation, such as potential limitations and additional considerations for using methods such as human judgments. This underscores additional considerations for and potential challenges of selecting data from two styles that may have underlying influences on how datasets are constructed.

Context-Dependence of Variation Relatedly, contextual considerations come into play with respect to the the Shakespeare to Modern English style transfer task, a dataset also reflective of transfer across stable, contextual boundaries. The Shakespeare to Modern English transfer task can be considered as transferring across temporal context, or as the characteristic style of a single author (Xu, 2017). In this case, while an influence of sociocultural context is apparent when considering the original data sources, the targeted stylistic variation occurs across such context boundaries. Thus, source of variation for textual features arising from external context lies with whether the intent is present for a dataset to represent a transfer across context boundaries, rather than an artifact reflecting specifics of dataset creation. This is illustrated in Figure 1 as a dashed line connecting style type to dataset properties.

With further regard to dataset creation, it is important to acknowledge that while we consider many properties arising from social influences as dynamic and variable influences giving rise to particular *styles*, a dataset will indirectly and inadvertently reflect such social context during creation to some degree. As such, we also must consider social factors **not** related to the actual targeted style, but rather arising from the dataset creation process. As an example of this consideration, we can't simply say two sentiment datasets from the same general domain (such as restaurant reviews) are equivalent if one was constructed with reviewers who had anonymity (in a sense mitigating some of the direct social pressure or influence) and the other was constructed with reviewers who were not anonymous and were thus subject to increased social pressure. By understanding both data and style differences and their interactions within a particular context, these potential differences or hidden influences can be more easily identified. In summary, the interactions between style and data properties are complex. While we have suggested interactions between context and sources of influence, there are likely correlations that exist based on sources of variation which future work can investigate.

5 Influences and Applications

In the previous sections, we demonstrated visible influences of style and dataset properties on performance, categorized a set of style and dataset properties for consideration, and discussed the potential interactions between sources of variation. We conclude by discussing several applications of understanding the sources of variation in text style transfer datasets. Specifically, we look at multi-task learning, domain adaptation, and generalizability.

Multi-Task Learning and Domain Adaptation Multi-task learning aims to jointly train a model with auxiliary tasks to complement learning of the target task. When determining which auxiliary objectives to incorporate, multi-task learning for

various NLP tasks has been shown to benefit from knowledge about both *dataset characteristics* and *stylistic properties*. For example, multi-task learning performance gains for NLP tasks such as POS tagging and text classification are predictable from dataset characteristics (Kerinec et al., 2018; Bingel and Søgaard, 2017). With regard to stylistic properties, within the context of multi-task learning for style transfer Zhang et al. (2020) achieved performance gains by leveraging an intuitive stylistic connection between formality data and grammatical error correction data.[11].

While multi-task learning can be viewed as a form of parallel transfer learning, we can view domain adaptation as a form of sequential transfer learning and look at similar applications of contextualizing stylistic variation. Li et al. (2019) found that leveraging generic style and content information outperformed generic content information alone for domain adaptation, however, the closeness of sentiment information (target attribute) in the source and target domains impacted performance. In other words how the style was reflected in the particular dataset (i.e., a dataset characteristic) was related to the benefit provided by the adaptation. Based on the combined evidence in this section, we can thus support applying analysis of both style and dataset properties for transfer learning data selection, including multi-task learning and domain adaptation, in text style transfer. We suggest that the taxonomy presented in this paper can assist exploration of systematic data selection methods in these and related application areas.

Generalizability One of the underlying motivations for pursuing multi-task learning and domain adaptation is the issue of generalizability. In the context of style transfer, we can consider generalizing a model for one style across different data distributions with the same stylistic attribute, or across similar domains yet different stylistic attributes. In either case, how the model learns to represent the generic style or content information is vital for successful transfer. As we've demonstrated throughout prior sections, considering both style and dataset properties can aid in identifying sources from which possible issues may arise in terms of along which dimensions stylistic attributes may significantly differ, or which artifacts or influences of dataset creation may influence generalizability secondary to any stylistic considerations. Considerations to this end may prove beneficial both in the dataset creation process as well as when considering how a model may perform beyond a specific dataset.

[11]Other styles, such as impoliteness and offense, are also highly dependent on each other (Kang and Hovy, 2021)

6 Conclusion

In this paper, we conducted a set of exploratory analyses to assess the visibility or influence of both style and dataset characteristics on text style transfer. Based on these observations, we proposed a categorization of stylistic and dataset properties that can contribute to variation in text style transfer datasets and described the applications in which these properties may be influential, limiting, or leveragable.

Acknowledgements

We thank the anonymous reviewers for their helpful comments and suggestions. We also thank Diyi Yang and Jingfeng Yang for a series of helpful discussions.

References

Amjad Abu-Jbara, Barbara Rosario, and Kent Lyons. 2011. Towards style transformation from written-style to audio-style. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 248–253, Portland, Oregon, USA. Association for Computational Linguistics.

Emily M Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Douglas Biber. 2012. Register as a predictor of linguistic variation. *Corpus linguistics and linguistic theory*, 8(1):9–37.

Douglas Biber and Susan Conrad. 2019. *Register, genre, and style*. Cambridge University Press.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169, Valencia, Spain. Association for Computational Linguistics.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 440–447, Prague, Czech Republic. Association for Computational Linguistics.

Eleftheria Briakou, Sweta Agrawal, Ke Zhang, Joel Tetreault, and Marine Carpuat. 2021. A review of human evaluation for style transfer. *arXiv preprint arXiv:2106.04747*.

Penelope Brown, Stephen C Levinson, and Stephen C Levinson. 1987. *Politeness: Some universals in language usage*, volume 4. Cambridge university press.

Yixin Cao, Ruihao Shui, Liangming Pan, Min-Yen Kan, Zhiyuan Liu, and Tat-Seng Chua. 2020. Expertise style transfer: A new task towards better communication between experts and laymen. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1061–1071, Online. Association for Computational Linguistics.

David Crystal and Derek Davy. 1969. *Investigating English Style*. Indiana University Press, Bloomington & London.

Yao Fu, Hao Zhou, Jiaze Chen, and Lei Li. 2019. Rethinking text attribute transfer: A lexical analysis. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 24–33, Tokyo, Japan. Association for Computational Linguistics.

Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and Li Deng. 2017. Stylenet: Generating attractive visual captions with styles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3137–3146.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1161–1166, Hong Kong, China. Association for Computational Linguistics.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, volume 1, pages 517–520. IEEE Computer Society.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Michael Alexander Kirkwood Halliday and Christian MIM Matthiessen. 2013. *Halliday's introduction to functional grammar*. Routledge.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 507–517, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Janet Holmes and Nick Wilson. 2017. *An introduction to sociolinguistics*. Routledge.

Dirk Hovy. 2018. The social and the neural network: How to make natural language processing about people again. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*, pages 42–49, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Dirk Hovy, Federico Bianchi, and Tommaso Fornaciari. 2020. "you sound just like your father" commercial machine translation systems include stylistic biases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1686–1690, Online. Association for Computational Linguistics.

Dirk Hovy and Diyi Yang. 2021. The importance of modeling social factors of language: Theory and practice. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 588–602, Online. Association for Computational Linguistics.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*, pages 10–19, Copenhagen, Denmark. Association for Computational Linguistics.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Florence, Italy. Association for Computational Linguistics.

Dongyeop Kang, Varun Gangal, and Eduard Hovy. 2019. (male, bachelor) and (female, Ph.D) have different connotations: Parallelly annotated stylistic language dataset with multiple personas. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1696–1706, Hong Kong, China. Association for Computational Linguistics.

Dongyeop Kang and Eduard Hovy. 2021. Style is NOT a single variable: Case studies for cross-stylistic language understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2376–2387, Online. Association for Computational Linguistics.

Hadi Kazemi, Seyed Mehdi Iranmanesh, and Nasser Nasrabadi. 2019. Style and content disentanglement in generative adversarial networks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 848–856. IEEE.

Emma Kerinec, Chloé Braud, and Anders Søgaard. 2018. When does deep multi-task learning work for loosely related document classification tasks? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 1–8, Brussels, Belgium. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Dianqi Li, Yizhe Zhang, Zhe Gan, Yu Cheng, Chris Brockett, Bill Dolan, and Ming-Ting Sun. 2019. Domain adaptive text style transfer. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3304–3313, Hong Kong, China. Association for Computational Linguistics.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921, Denver, Colorado. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97.

Remi Mir, Bjarke Felbo, Nick Obradovich, and Iyad Rahwan. 2019. Evaluating style transfer for text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 495–504, Minneapolis, Minnesota. Association for Computational Linguistics.

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

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

Kelly Peterson, Matt Hohensee, and Fei Xia. 2011. Email formality in the workplace: A case study on the Enron corpus. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 86–95, Portland, Oregon. Association for Computational Linguistics.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876, Melbourne, Australia. Association for Computational Linguistics.

Reid Pryzant, Richard Diehl Martinez, Nathan Dass, Sadao Kurohashi, Dan Jurafsky, and Diyi Yang. 2020. Automatically neutralizing subjective bias in text. In *Proceedings of the aaai conference on artificial intelligence*, volume 34, pages 480–489.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 129–140, New Orleans, Louisiana. Association for Computational Linguistics.

Melissa Roemmele, Andrew S Gordon, and Reid Swanson. 2017. Evaluating story generation systems using automated linguistic analyses. In *SIGKDD 2017 Workshop on Machine Learning for Creativity*, pages 13–17.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with Bayesian optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 372–382, Copenhagen, Denmark. Association for Computational Linguistics.

Cicero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. 2018. Fighting offensive language on social media with unsupervised text style transfer. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 189–194, Melbourne, Australia. Association for Computational Linguistics.

Stephanie Schoch, Diyi Yang, and Yangfeng Ji. 2020. "This is a problem, don't you agree?" Framing and bias in human evaluation for natural language generation. In *Proceedings of the 1st Workshop on Evaluating NLG Evaluation*, pages 10–16, Online (Dublin, Ireland). Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6833–6844.

Anders Søgaard, Barbara Plank, and Dirk Hovy. 2014. Selection bias, label bias, and bias in ground truth. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Tutorial Abstracts*, pages 11–13, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

William Strunk and E. B. White. 1999. *The elements of style*, 4th ed edition. Allyn and Bacon, Boston.

Shaolei Wang, Wangxiang Che, Qi Liu, Pengda Qin, Ting Liu, and William Yang Wang. 2020. Multi-task self-supervised learning for disfluency detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9193–9200.

Yue Wang, Cuong Hoang, and Marcello Federico. 2021. Towards modeling the style of translators in neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1193–1199.

Yunli Wang, Yu Wu, Lili Mou, Zhoujun Li, and Wenhan Chao. 2019. Harnessing pre-trained neural networks with rules for formality style transfer. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3573–3578, Hong Kong, China. Association for Computational Linguistics.

Joseph M. Williams and Joseph Bizup. 2017. *Style: lessons in clarity and grace*, twelfth edition edition. Pearson, Boston.

Wei Xu. 2017. From shakespeare to Twitter: What are language styles all about? In *Proceedings of the Workshop on Stylistic Variation*, pages 1–9, Copenhagen, Denmark. Association for Computational Linguistics.

Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. In *Proceedings of COLING 2012*, pages 2899–2914, Mumbai, India. The COLING 2012 Organizing Committee.

Yi Zhang, Tao Ge, and Xu Sun. 2020. Parallel data augmentation for formality style transfer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3221–3228, Online. Association for Computational Linguistics.

A Dataset Details

We selected English text style datasets with a single transferred stylistic attribute between two classes. Of importance for inclusions were datasets that exhibited different creation methods: both automatically annotated and human annotated. Where available, we used the original (or pre-existing, as with the case of the Shakespeare dataset) train/val/test data splits. Links to each dataset are provided through the respective citations.

Fluency Contains aligned sentence pairs labeled as fluent or disfluent, from the English Switchboard (SWBD) Corpus (Godfrey et al., 1992; Wang et al., 2020). Train/val/test split: 173.7k/10.1k/7.9k

GYAFC-EM & GYAFC-FR Contain aligned sentence pairs labeled as informal or formal, from the *Entertainment & Music* and *Family & Relationships* domains, respectively, of the question answering forum Yahoo Answers (Rao and Tetreault, 2018). GYAFC-EM & GYAFC-FR datasets can be requested at https://github.com/raosudha89/GYAFC-corpus. GYAFC-EM Train/val/test split: 52.6k/2.9k/1.4k; GYAFC-FR Train/val/test split: 52k/2.8k/1.3k

Biased-Word Contains aligned sentence pairs labeled as subjective or neutral, crawled from 423,823 Wikipedia editor neutrilization revisions between 2004 and 2019 (Pryzant et al., 2020). Train/val/test split: 53.8k/700/1k

Flickr Contains sentence pairs captioning an image, labeled as romantic or humorous (Gan et al., 2017). We created a 6k/500/500 Train/val/test split since only the original 7k training instances are available.

Shakespeare Contains sentence pairs labeled as Shakespeare or modern English (Xu et al., 2012). Sentences are crawled from 17 Shakespeare plays from Sparknotes [12], which provides the modern counterparts. Following Jhamtani et al. (2017), we use 15 plays for training, with *Twelfth Night* used for validation, and *Romeo and Juliet* used for testing.

B Similarity Metrics

In Table 3 we do not distinguish between source and target direction due to the symmetry of metrics in our setting. We provide further justification below:

Jaccard similarity can be defined as

$$\frac{\mathcal{V}_{\{\boldsymbol{s}^{(k)}\}} \cap \mathcal{V}_{\{\boldsymbol{t}^{(k)}\}}}{\mathcal{V}_{\{\boldsymbol{s}^{(k)}\}} \cup \mathcal{V}_{\{\boldsymbol{t}^{(k)}\}}} \tag{1}$$

where $\mathcal{V}_{\{\boldsymbol{s}^{(k)}\}}$ denotes the set of vocabulary words existing in a source sentence $\{\boldsymbol{s}^{(k)}\}$ and $\mathcal{V}_{\{\boldsymbol{t}^{(k)}\}}$ denotes the set of vocabulary words existing in a target sentence $\{\boldsymbol{t}^{(k)}\}$. By the commutative property, $\mathcal{V}_{\{\boldsymbol{s}^{(k)}\}} \cap \mathcal{V}_{\{\boldsymbol{t}^{(k)}\}} = \mathcal{V}_{\{\boldsymbol{t}^{(k)}\}} \cap \mathcal{V}_{\{\boldsymbol{s}^{(k)}\}}$ and $\mathcal{V}_{\{\boldsymbol{s}^{(k)}\}} \cup \mathcal{V}_{\{\boldsymbol{t}^{(k)}\}} = \mathcal{V}_{\{\boldsymbol{t}^{(k)}\}} \cup \mathcal{V}_{\{\boldsymbol{s}^{(k)}\}}$, making Jaccard similarity symmetric. Word-based Levenshtein distance is defined as the minimum number of edit operations to convert $\{\boldsymbol{s}^{(k)}\}$ to $\{\boldsymbol{t}^{(k)}\}$ through insertions, deletions, and substitutions. Substitutions are symmetric by definition, and insert and delete operations to convert $\{\boldsymbol{s}^{(k)}\}$ to $\{\boldsymbol{t}^{(k)}\}$ are simply reversed when converting $\{\boldsymbol{t}^{(k)}\}$ to $\{\boldsymbol{s}^{(k)}\}$. In $LD_{norm}(s,t)$, we normalize by $\max|s|,|t|$, which is invariant to order. Finally,

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{2}$$

where $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$.

In our setting, $TP = w \in s \cap t$, $FP = w \in \boldsymbol{s} \backslash \boldsymbol{t}$, and $FN = w \in \boldsymbol{t} \backslash \boldsymbol{s}$. By these definitions, *FP* and *FN* are reversed when source and target are reversed, and therefore by definition, F_1 is symmetric when comparing source and target sentence pairs.[13]

C Linguistic Features

Lexical Complexity Lexical complexity refers to the complexity of words based on the length or number of syllables. We use average word length in characters (Pavlick and Tetreault, 2016) and average number of syllables, with and without stopwords.

Lexical Diversity Size of vocabulary has been used as a feature for style categorization in prior work (Abu-Jbara et al., 2011). We chose to include unigrams and bigrams to reflect diversity of vocabulary as well as diversity of expression.

[12] https://www.sparknotes.com/

[13] Acronyms refer to "True Positives" (TP), "False Positives" (FP), and "False Negatives" (FN). We consider target as ground truth and copy source over as a "generated" target. We essentially consider positives as words that are generated and negatives as words that are not generated, with truth values corresponding to whether or not a word *should* have been generated.

POS Tags POS tags have been used extensively in the stylistic analysis of text, including formality (Pavlick and Tetreault, 2016) and written-style vs. audio-style (Abu-Jbara et al., 2011). Granularity of POS tags has stylistic implications, such as implications for different specific punctuation types (Strunk and White, 1999), so we include Universal and Treebank POS tags for course-grained and fine-grained stylistic information, respectively. [14]

Both Universal and Treebank POS tags are processed using Stanza (Qi et al., 2020), which correspond with the Universal Dependencies (McDonald et al., 2013) POS tags and the Penn Treebank (Marcus et al., 1993) English POS tagset.

Sentence Length Sentence length has stylistic implications (Strunk and White, 1999) and has been used as a feature to classify various styles, such as written-style and audio style (Abu-Jbara et al., 2011) and formality (Pavlick and Tetreault, 2016). We include sentence length in words and sentence length in tokens to account for punctuation differences.

Phrases Measures of phrases and clauses have been used for stylistic analysis in terms of syntactic complexity (Abu-Jbara et al., 2011). We include measures of noun phrases, verb phrases, and dependent clauses.

Readability We adopt the readability measures Flesch-Kincaid Grade Level score (Pavlick and Tetreault, 2016) and ratio of complex words (Abu-Jbara et al., 2011) from prior studies.

Subjectivity We adopted several measures of subjectivity from Pavlick and Tetreault (2016) and adapted the measure ratio of pronouns (Abu-Jbara et al., 2011) by measuring the individual type counts of 1st, 2nd, and 3rd person pronouns.

Bag-of-Words We include the bag-of-words feature to account for cross-class vocabulary differences.

D Jensen-Shannon Divergence

While we indicate large Jensen-Shannon Divergences in Table 5, we include the full range of Jensen-Shannon Divergence results in Table 6 in a numerical format as well.

	Flick	Shake	GY-FR	GY-EM	Bias	Flu.
FF	0.022	0.019	0.019	0.027	0.003	0.004
LexC	**0.086**	0.039	**0.132**	0.054	0.047	0.004
Read	**0.081**	0.040	**0.079**	0.056	0.050	0.013
LexD	0.067	0.049	0.041	0.050	0.031	**0.108**
UPOS	**0.088**	0.052	0.066	**0.075**	0.034	0.011
XPOS	0.063	0.042	0.052	0.056	0.026	0.008
SenL	**0.137**	**0.090**	0.070	0.062	0.013	0.017
Phr	**0.105**	0.056	0.064	0.065	0.030	0.024
Sub	**0.107**	**0.075**	0.054	0.057	0.064	0.016
BoW	0.018	0.015	0.013	0.011	0.002	0.002

Table 6: Jensen-Shannon divergence between source and target on each test set using feature groupings in Table 4. Scores ≥ 0.075 are made bold.

[14] Although we used state-of-the-art tools to extract features such as part-of-speech tags, we do note the possibility of tool performance differences across datasets (Søgaard et al., 2014). However, as we utilize the same tool for both the classification and ablation study as well as the divergence scores, we expect the impact of tool performance within a dataset to have minimal impact on resulting conclusions.

Generation Challenges: Results of the Accuracy Evaluation Shared Task

Craig Thomson
Dept of Computing Science
University of Aberdeen
Aberdeen, UK
c.thomson@abdn.ac.uk

Ehud Reiter
Dept of Computing Science
University of Aberdeen
Aberdeen, UK
e.reiter@abdn.ac.uk

Abstract

The Shared Task on Evaluating Accuracy focused on techniques (both manual and automatic) for evaluating the factual accuracy of texts produced by neural NLG systems, in a sports-reporting domain. Four teams submitted evaluation techniques for this task, using very different approaches and techniques. The best-performing submissions did encouragingly well at this difficult task. However, all automatic submissions struggled to detect factual errors which are semantically or pragmatically complex (for example, based on incorrect computation or inference).

1 Introduction

Users expect data-to-text natural language generation (NLG) systems to generate textual summaries which are accurate. However, many NLG systems, especially neural ones, generate texts which are factually incorrect.

The most reliable way to assess the accuracy of a generated text is to ask human annotators to carefully fact-check the text. However this is a time-consuming and expensive process. In earlier work, we developed a protocol (Thomson and Reiter, 2020) where three Mechanical Turk workers (who had been screened and passed a qualifying test) carefully annotated factual errors in a text produced by a neural NLG system. The protocol was effective and showed high interannotator agreement, but it took annotators 20-30 minutes (each) to fact-check a moderately complex 300-word paragraph produced by a neural data-to-text NLG system. The total cost of the process (including fees to Amazon and money spent on the screening process for potential annotators) was about US$30 per text.

It would be very useful to the NLG community if we could come up with quicker and easier ways of measuring accuracy and factual correctness which have good correlations with the protocol of Thomson and Reiter (2020). Such methods could be based on less time-consuming human evaluations or on automatic metrics. However, these techniques should only be used if they have good agreement and correlation with careful high-quality human fact-checking by multiple annotators.

In this shared task, participating teams submitted techniques (both human and automatic) for evaluating the factual accuracy of summaries of basketball games produced from box score (and other game data) by three neural NLG systems. These techniques were evaluated by computing precision and recall (of identified factual errors) against a gold-standard human annotation produced by Thomson and Reiter (2020)'s protocol. Some of the systems did well overall, but it was also clear that some types of factual errors are difficult to detect.

We hope that our shared task encourages researchers from many fields to work on the problem of identifying factual errors in generated texts; progress in this area would be very helpful for NLG. Full details of the shared task requirements, as well as both the training and test corpus can be found at https://github.com/ehudreiter/accuracySharedTask.

2 Task

Participants were asked to submit a technique for identifying incorrect statements in a generated text. This meant statements which are not true in the real world (ie, classic fact-checking), not just statements which disagree with (or are not derivable from) the system run-time data (see Section 3.1 of Thomson and Reiter (2020)). Techniques could be

- Human evaluation protocols. Subjects would have access to data about the game and the teams, and also (if part of the protocol) to a human-authored reference text.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 240–248,
Aberdeen, Scotland, UK, 20-24 September 2021.

- Automatic metric (algorithm). The algorithm will have access to data about the game and the teams, and to a reference text.

- A combination of human evaluation and automatic metrics.

The output of the evaluation protocol or metric was a list of mistakes in the text. Each mistake was characterised by

- Its position in the text (start token and end token).

- A category. We use the following categories, which are based on Thomson and Reiter (2020)
  - *Incorrect number:* It does not matter whether the number is spelled out or is in digits.
  - *Incorrect name (for named entities):* In a sports reporting context, this includes people, places, teams, and days of the week.
  - *Incorrect word:* A word which is not one of the above and is incorrect.
  - *Context error:* A phrase which causes an incorrect inference because of context or discourse.
  - *Not checkable:* A statement which can not be checked, either because the information is not available or because it is too time-consuming to check.
  - *Other:* Any other type of mistake.

An example is shown in Figure 1. Note that this example combines fragments from texts produced by several different systems, along with some manual adjustments, in order to illustrate different types of mistakes in a simple way.

3 Data

We manually annotated, using the procedure of Thomson and Reiter (2020), 90 texts produced by three neural NLG systems that use basketball box score data: Wiseman et al. (2017), Puduppully et al. (2019a), and Rebuffel et al. (2020). In total, 30 texts were annotated from each system. Of these, 60 texts (20 from each system) were given to shared task participants as training data, and 30 texts (10 from each system) were reserved for a separate test set, which participants did not see until they had submitted their solutions.

Annotators were recruited on the Amazon Me-

The Memphis Grizzlies (5-2) defeated the Phoenix Suns (3 - 2) Monday 102-91 at the Talking Stick Resort Arena in Phoenix. The Grizzlies had a strong first half where they out-scored the Suns 59-42. Marc Gasol scored 18 points, leading the Grizzlies. Isaiah Thomas added 15 points, he is averaging 19 points on the season so far.

List of errors:

- 2: incorrect number, should be 0.
- Monday: incorrect named entity, should be Wednesday.
- Talking Stick Resort Arena: incorrect named entity, should be US Airways Center.
- strong: incorrect word, the Grizzlies did not do well in the first half.
- out-scored: incorrect word, the Suns had a higher score in first half.
- 59: incorrect number, should be 46.
- 42: incorrect number, should be 52 .
- leading: incorrect word. Marc Gasol did not lead the Grizzlies, Mike Conley did with 24 points.
- Isaiah Thomas added: context error. Thomas played for the Suns, but context here implies he played for the Grizzlies and added to their score.
- averaging 10 points on the season so far: not checkable. This is very hard to check, since data sources report performance per season and per game, not performance up to a particular point in a season.

Figure 1: Example text with error annotations. Corrections and explanations are not required, but are included here for clarity. Box score data for this game is available at https://www.basketball-reference.com/boxscores/201411050PHO.html .

---

chanical Turk platform. Fair treatment and compensation of workers is essential (Silberman et al., 2018), not only from an ethical standpoint, but to ensure high quality annotations. We paid annotators approximately US$20 per hour. The same three annotators marked up all 90 texts.

3.1 Systems Used

The three neural systems we used explored different ways of modifying the neural architecture. The system of Wiseman et al. (2017) defined the Rotowire task and provided initial benchmarks for machine translation systems using copy attention, it is included for this reason. Puduppully et al. (2019a) learned a document plan which was then used to generate text, whilst Rebuffel et al. (2020) used a hierarchical encoder to group attributes (such as statistics) by their respective entities (players/teams).

Other systems in this domain which could be used for evaluation include Puduppully et al. (2019b), Wang (2019), Gong et al. (2019), and Iso et al. (2019). Our aim, however, is to assess how well results produced by the participant's evaluation techniques correlate with the gold-standard fact-checking. Hence we are looking for a set of systems which generate texts that contain a significant number of accuracy errors, not complete coverage of all systems that generate texts from basketball box score data.

3.2 Multiple Correct Annotations

Sometimes there are multiple correct ways of annotating errors. For example, consider the sentence

> Lou Williams led the team in scoring, dropping 30 points, six rebounds and seven assists

Suppose that it was another player, Solomon Hill, who had 30 points, 6 rebounds, and 7 assists. In this case, the sentence could be corrected either by changing the player name (to Solomon Hill), or by changing the statistics (to the correct ones for Lou Williams). In such cases we asked annotators to try to find the smallest number of annotations required to correct the sentence, prioritising categories in the order of Name, Number, Word, Context, Other, Not checkable. This is straightforward in this example. where the choice is correcting a single player name, or three numbers.

There were, however, a few cases where multiple complex annotations were plausible and the preferred one was not clear to our annotators. For example, in our test we encountered a sentence that was marked up by annotators as shown in Figure 2:

> **Annotator T1**: The only other Raptor to reach double figures in points was Dwyane Dragic, who came off the bench for 22 points (9-17 FG, 3-7 3Pt, 3-3 FT), six rebounds and five assists.
>
> **Annotator T2**: The only other Raptor to reach double figures in points was Dwyane Dragic, who came off the bench for 22 points (9-17 FG, 3-7 3Pt, 3-3 FT), six rebounds and five assists.
>
> **Annotator T3**: The only other Raptor to reach double figures in points was Dwyane Dragic, who came off the bench for 22 points (9-17 FG, 3-7 3Pt, 3-3 FT), six rebounds and five assists.

Figure 2: Annotations by each annotator, showing Name, Number, and Word errors.

T1 and T2 essentially decided to change the player name to *Goran Dragic*; since *Dragic* played for the other team (*Heat*), they also corrected *Raptors*. They then corrected three of the numbers accordingly and noted that *Dragic* did not come off the bench, he started the game. T3 disagreed, changing the player name to *Lou Williams* who did in fact start for the *Raptors*. Whilst this minimised Name and Word errors, it required correcting 7 of the numbers, leading to 9 errors in all, compared to the 7 errors annotated by T1 and T2.

The majority annotation (T1 and T2) was correct in this case according to our 'choose annotation with smallest number of errors'. But it is not trivial for annotators to search through multiple possible annotations looking for the optimal one, and in a larger sense it is not clear which annotation is 'correct'.

4 Accuracy Errors Observed

In this section we discuss and give some insights about the accuracy errors we observed in the manually-annotated training data (i.e, the 60 annotated texts given to participants as training data). We look separately at the different types of errors listed in section 2, and also at the impact of position

Error	Type	count	note
NUM-DIGIT	Number	270	number in digits, such as an incorrect quantity of points
TEAM	Name	162	name of team, such as *Miami Heat*
NUM-WORD	Number	130	a number spelled as a word or words
DAY-WEEK	Name	128	a day of the week, such as *Wednesday*
PLAYER	Context	50	player name (used in incorrect context)
led	Word	40	word *led*, often indicates a player led their team by some measure
a (an)	Number	34	*a* or *an* meaning the number 1
ORDINAL	Number	26	ordinal number often describing consecutive games
double-double	Word	23	word *double-double*, a basketball metric
PLAYER	Name	21	name of a player, such as *LeBron James*

Table 1: Errors that occurred at least 20 times in the training data. NUM-DIGIT, TEAM, NUM-WORD, DAY-WEEK, ORDINAL refer to types of words. Number, Name, Context, Word refer to types of errors.

and the neural NLG system used. Table 1 lists all errors that occurred at least 20 times in the training data.

4.1 Number errors

Number errors are the most common type of error in our corpus; there were 474 Number errors in the 60 texts in the training data. This category includes errors in numbers presented as digits (NUM-DIGIT), errors in spell-out numbers (NUM-WORD), and errors when a/an is used to mean the number 1.

From a semantic perspective, we can distinguish between *errors in copying numbers from the data* (eg, claiming that Smith scored 20 points when the box score data says that he scored 10 points) and *errors in calculating numbers which are not directly in the data* (eg, calculating the score at half-time, from the quarter-level scores given in the box office data). Both types of errors were common in our corpus.

4.2 Name errors

There were 317 Name errors in our corpus. TEAM, PLAYER, and DAY-WEEK (from Table 1) are all examples of a Name error. Many of these errors arose when NLG systems tried to create sentences for which they lacked data, such as the following:

> The Sixers' next game will be at **home** against the **New Orleans Pelicans** on **Wednesday**

Information about the next game is not present in the data used by the three systems which were fact-checked, so they simply guessed team and day of week, and usually guessed wrong. Of course we cannot expect a system to generate accurate texts that communicate information which is not present in the input data! But we can expect data-to-text systems to avoid sentences which communicate unavailable data.

As mentioned in subsection 3.2, sometimes a sentence could be characterised as having either a Name or a Number error. In such cases we asked annotators to make the correction which required the smallest number of changes.

4.3 Word errors

There were 334 Word errors in our corpus. They can be divided into two categories: errors in using words with clear unambiguous definitions (such as *out-scored* in Figure 1) and errors in words with fuzzy definitions (such as *strong* in Figure 1).

The most common error in a well-defined word is *double-double*. A double-double occurs when a player has ten or more (double-digits) in exactly two of the following categories: points, rebounds, assists, steals, and blocks. Note that if a player has ten or more in three of the categories, this is called a *triple-double* (3 statistics in double-digits) rather than a *double-double*. While double-double is easy to define via rules, there were 23 mistakes in our 60 corpus texts (Table 1) in the usage of this word; this seems to be a difficult concept for our systems to learn.

The most common error in a fuzzy word was *led*. *Led* appears in many contexts, for example we can say that a player *led the bench in scoring* or that a team *led at the half*.

The meaning of *led* is not clear-cut, and indeed on a few occasions the annotators disagreed on whether *led* was appropriate. This is because *led* (when used in descriptions of basketball games) can encompass rebounds, assists, steals and blocks

as well as points. For example, if one player has 25 points, 0 assists and 0 rebounds, and a second player has 22 points, 10 assists, and 10 rebounds, then the first player led in scoring, but it could be argued that the second player had a stronger performance overall, and therefore *led*. However, most of the incorrect usages of *led* marked by the annotators were in cases where all of the annotators agreed that *led* was inappropriate.

Some *ORDINAL* errors were related to this. For example, one sentence would state that a player *led*, and the subsequent sentence would state that a second player *was second*.

4.4 Context error

A Context occur occurs when a statement is literally true but misleading in context. There were 51 Context errors in our corpus, 50 of which involved PLAYERs. Typically the text would mislead the reader as to a player's status, especially which team he is playing for. An example from Figure 1 is:

> Marc Gasol scored 18 points, leading the Grizzlies. Isaiah Thomas added 15 points

Thomas scored 15 points but played for the other team (Suns). This is a Context error, since the context implies that Thomas played for the Grizzlies.

Such errors were common, the systems had a difficult time in learning when it is contextually appropriate to mention a person.

4.5 Not Checkable and Other

A Not Checkable error occurs when the annotator cannot check whether a fact is true or not. There were 37 such errors in our corpus. They usually occurred when complex statistics were given which were difficult and time-consuming for annotators to check. In order to keep the annotation task manageable, annotators were told not to look at more than 4 previous games. This made it impossible to check statements such as *he is averaging 19 points on the season so far* (from Figure 1), which requires looking at data from every previous game in the season.

We discouraged our annotators from using the Other category unless absolutely necessary, and in fact there was only one Other error in our corpus, which was the nonsensical statement *run with the first - round win of the season*.

4.6 Position analysis

In addition to analysing errors by category, we also wondered if there might be fewer errors at the beginning of the text, and more at the end. There was in fact a sharp increase in Name errors in the last sentence (Figure 3), but this was probably due to the fact that the last sentence usually described next games, and the systems did not have this information so they hallucinated. Other than this, we did not see an increase in errors later in the text. Figure 4 shows the distribution of Number errors in different positions, the other error types (excluding Name) have a similar distribution. For both of these figures, error counts are shown based upon which tenth of the summary (by token id) the error starts in.

4.7 System analysis

Last but not least, we wanted to look at the error profiles for the three systems we included in the error corpus. Two of the systems used RNN-based encoders (Wiseman et al., 2017; Puduppully et al., 2019a) and the third used a hierarchical transformer (Rebuffel et al., 2020). Table 2 shows the errors each system had within each category. The hierarchical transformer made fewer Number errors than both RNN based systems but more Context errors. It is unclear why the hierarchical encoder of (Rebuffel et al., 2020) made more Context errors, although it may be learning to better group entities with their attributes, at the expense of ordering between entities.

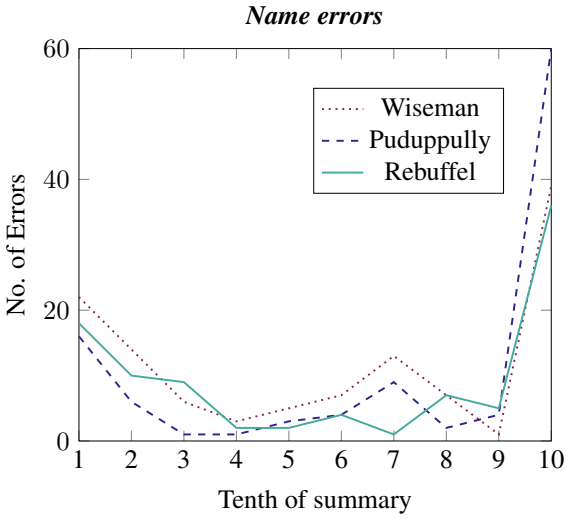


Figure 3: Name errors in different tenths of the summary.

System	encoder	NAME	NUMBER	WORD	CONTEXT	NOT_CHECK	OTHER
Wiseman	RNN	5.9	10.4	6.7	0.3	1.0	0.0
Puduppully	RNN	5.3	7.9	5.1	0.6	0.4	0.0
Rebuffel	transformer	4.7	5.5	5.0	1.7	0.5	0.1

Table 2: Breakdown of error types per-text, by system. 20 texts were included in the training corpus for each system.

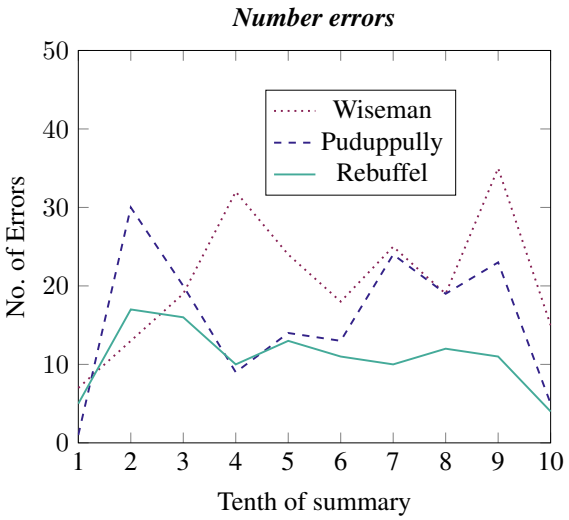


Figure 4: Number errors in different tenths of the summary.

5 Submissions

5.1 Automatic approaches

5.1.1 Charles-UPF

Charles University and Pompeu Fabra University submitted a system which detects errors using a three-stage process

1. A rule-based NLG system is used to generate sentences with facts that can be derived from the game data.

2. For each sentence in the NLG texts, a subset of the sentences in (1) is chosen based on semantic similarity to the target sentence.

3. A language model is used to identify errors. The input to the model is both the target sentence and the sentences in (2). The model is trained on synthetic data as well as the training data.

Note that the Charles-UPF system checks sentences separately, so it cannot detect errors that depend on document-level context, including Context errors and usage of '**only other**' (subsection 6.1).

5.1.2 Eurecom

The Eurecom system follows an approach inspired by earlier work on computational fact-checking (Karagiannis et al., 2020). It focuses on identifying Number errrors, and also Word errors where the word maps in a straightforward way to the game data, such as errors in the usage of '**defeated**'. A three-step process is used

1. *Claim identification:* Factual claims are extracted from the NLG text.

2. *Property identification*: The claims in (1) are expanded into full property specifications; for example the claim *18 points* is expanded with the name of the player who is supposed to have scored these points.

3. *Claim verification*: The game data is queried using the property specifications in (2); incorrect claims are flagged.

5.1.3 National Institute of Japanese Literature

The NIJL system used different approaches for different types of errors:

- *Word and Name errors*: A set of rules was used to identify Word and Name errors in the NLG texts. These rules were tuned to the structure of game summaries, with different rule used for lead, middle, and end sections of the summaries. The rules referred to the human reference texts as well as the game data.

- *Number errors*: A classifier was used to predict what relation each number represented. A co-reference tool was used to resolve referring expressions such as '*he*'.

The NIJL system was the only submission which used the human-written reference texts as well as game data when looking for accuracy errors; all other submissions just used the game data.

5.2 Hybrid approaches

5.2.1 Laval University

The Laval University approach was a hybrid system, which combined automatic analysis and human annotation.

1. *Pre-annotation*: a set of rules and classifiers are used to highlight potential accuracy errors in the NLG text. Row-column lookup on source data is used to identify potential Name and Number errors, and a multi-class, multi-label classifier is trained for Word, Context, and Not Checkable errors.

2. *Human annotation*: a single human annotator then annotated errors in the NLG text, using the pre-annotation of (1) to help them.

The human annotation was much quicker than the protocol of Thomson and Reiter (2020), because of the pre-annotation step.

We present two results for Laval University: a 'metric' result which is based purely on the results of the pre-annotation process, and a 'hybrid' result which is based on the full approach described above.

6 Results

The submissions were evaluated by computing their recall and precision against the gold-standard mistake list (GSML) which was based on the human annotated texts in the test set (section 3). In other words, for each submission, we calculated how many of the gold-standard mistakes were detected by that submission (recall), and how many of the mistakes detected by that submission were present in the gold-standard annotation (precision). We calculated this at the level of both mistakes and tokens.

Table 3 shows the recall and precision of our submissions against the gold-standard manually annotated texts, for the 30 texts in the test set. We can see that the Laval University hybrid approach did best. Amongst the automatic evaluations, the Charles-UPF system had the best recall and precision.

Tables 4 to 8 show recall/precision of the submissions for different types of mistakes, as well as overall. We can see that the automatic techniques (Tables 5 to 8) were unable to detect Context and Other errors, and only the Laval University (metric) system could detect Not Checkable errors (but at low precision and recall). We can also see that none of the automatic systems did well at detecting Word errors; the best system, Charles-UPF, had around 50% precision and recall. Overall, this suggests that semantically more complex errors are harder to detect automatically, which is not surprising.

As a point of comparison, the Relation Generation metric (Wiseman et al., 2017), which has been widely used by many previous papers to evaluate accuracy, can only detect Name and Number errors and has a recall of less than 40% for these types of errors (Thomson and Reiter, 2020). This is considerably worse than the best-performing submissions to our shared task.

	Mistake		Token	
Team	recall	precision	recall	precision
Laval University*	0.841	0.879	0.668	0.859
Charles-UPF	0.691	0.756	0.550	0.769
NIJL	0.523	0.494	0.349	0.505
Laval University	0.503	0.334	0.410	0.397
Eurecom	0.080	0.311	0.046	0.202

Table 3: Results of the Accuracy Evaluation Shared Task for all submissions. The * denotes the hybrid evaluation for Laval University. All other submissions were metrics.

	Mistake		Token	
Team	recall	precision	recall	precision
Name	0.920	0.938	0.929	0.919
Number	0.862	0.881	0.832	0.854
Word	0.679	0.731	0.561	0.685
Context	0.750	0.400	0.733	0.367
Not checkable	0.237	0.391	0.073	0.615
Other	0.000	-	0.000	-
Overall	0.841	0.879	0.668	0.859

Table 4: Laval University (hybrid evaluation) per-type results.

	Mistake		Token	
Team	recall	precision	recall	precision
Name	0.750	0.846	0.759	0.862
Number	0.777	0.750	0.759	0.752
Word	0.514	0.483	0.465	0.529
Context	0.000	-	0.000	-
Not checkable	0.000	-	0.000	-
Other	0.000	-	0.000	-
Overall	0.691	0.756	0.550	0.769

Table 5: Charles-UPF (metric) per-type results.

Team	Mistake		Token	
	recall	precision	recall	precision
Name	0.000	-	0.000	-
Number	0.205	0.329	0.198	0.203
Word	0.014	0.095	0.006	0.095
Context	0.000	-	0.000	-
Not checkable	0.000	-	0.000	-
Other	0.000	-	0.000	-
Overall	0.080	0.311	0.046	0.202

Table 6: Eurecom (metric) per-type results.

Team	Mistake		Token	
	recall	precision	recall	precision
Name	0.594	0.787	0.641	0.811
Number	0.442	0.351	0.427	0.340
Word	0.357	0.137	0.207	0.146
Context	0.000	-	0.000	-
Not checkable	0.500	0.190	0.200	0.407
Other	0.000	-	0.000	-
Overall	0.503	0.334	0.410	0.397

Table 7: Laval University (metric) per-type results.

6.1 Error analysis: The blind spot of metric submissions

To explore our intuition that complex errors were harder for the automatic systems to find, we performed a preliminary error analysis on the 84 mistakes (of 622) that were missed by all automatic submissions (the blind spot). We categorised each mistake based on the type of sentence that contained it:

Simple: Only 27 of the mistakes were simple, such as an incorrect attribute for an entity, or an incorrect name for a set of attributes. An example is '***Buddy Hield*** *led the second unit with a season - high 29 points , along with one assist , one rebound and one steal*', where the statistics belonged to Eric Gordon.

Comparison: 26 of the mistakes involved the comparison of how two teams fared in a quarter/half, or how their statistics compared in the game. An examples is '*The Nets got off to a quick start in this one, out-scoring the Kings 28-28 right away in the first quarter.*', where the tie of 28 points in the first quarter is incorrectly described. Many of these mistakes also involved getting the X-Y numbers wrong.

Only other: 14 of the mistakes were in clauses like '*The only other Net to reach double figures in points was Ben McLemore*'. This requires models

Team	Mistake		Token	
	recall	precision	recall	precision
Name	0.358	0.974	0.258	0.974
Number	0.696	0.419	0.672	0.419
Word	0.350	0.301	0.245	0.310
Context	0.000	-	0.000	-
Not checkable	0.000	-	0.000	-
Other	0.000	-	0.000	-
Overall	0.523	0.494	0.349	0.505

Table 8: National Institute of Japanese Literature (metric) per-type results.

and metrics to determine:

- That Ben McLemore had double-figures and was a Nets player.
- Which other Nets had double-figures.
- That all such players have been mentioned previously.

Data outwith game: 11 of the mistakes required data from outwith the game being summarised, including averages over prior games (8 mistakes), and the upcoming game schedule (3 mistakes).

Player groups: 6 mistakes incorrectly described a group of players, such as a duo.

45% of blind spot mistakes involved Word, Context, and Not-Checkable errors, compared to only 30% overall in the GSML. In addition, only 8% of blind spot mistakes were cardinal numbers, despite these constituting 33% of the GSML. It is important that we do not miss blind spot mistakes as whilst they are only 14% of the current GSML, this proportion could increase as systems become better at avoiding simple errors.

7 Conclusion

Neural data-to-text systems need to be able to produce accurate texts in order to be genuinely useful in most NLG use cases. An essential prerequisite to improving accuracy is being able to measure and evaluate accuracy.

We believe that the evaluation techniques submitted to our shared task represent a major advance in the state of the art. We encourage participants and others to continue developing better-performing techniques for this key evaluation task.

Acknowledgments

We are very grateful to all of the participants in the shared task, for their hard work in exploring very diverse approaches to the problem of finding accuracy errors! Several other teams were unable to complete a submission by our deadline; they had very interesting ideas and we encourage them to continue working on their ideas. We are also very grateful to Samira Shaikh and the members of the Aberdeen CLAN group for their help and advice. Last but not least, we are very grateful for the hard work of our Mechanical Turk annotators, in creating our training and test data. Craig Thomson's work is supported under an EPSRC NPIF studentship grant (EP/R512412/1).

References

Heng Gong, Xiaocheng Feng, Bing Qin, and Ting Liu. 2019. Table-to-text generation with effective hierarchical encoder on three dimensions (row, column and time). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3143–3152, Hong Kong, China. Association for Computational Linguistics.

Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoaki Okazaki, and Hiroya Takamura. 2019. Learning to select, track, and generate for data-to-text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2102–2113, Florence, Italy. Association for Computational Linguistics.

Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. 2020. Scrutinizer: A mixed-initiative approach to large-scale, data-driven claim verification. *Proc. VLDB Endow.*, 13(11):2508–2521.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019a. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019b. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy. Association for Computational Linguistics.

Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. 2020. A hierarchical model for data-to-text generation. In *Advances in Information Retrieval*, pages 65–80, Cham. Springer International Publishing.

M. S. Silberman, B. Tomlinson, R. LaPlante, J. Ross, L. Irani, and A. Zaldivar. 2018. Responsible research with crowds: Pay crowdworkers at least minimum wage. *Commun. ACM*, 61(3):39–41.

Craig Thomson and Ehud Reiter. 2020. A gold standard methodology for evaluating accuracy in data-to-text systems. In *Proceedings of INLG 2020*.

Hongmin Wang. 2019. Revisiting challenges in data-to-text generation with fact grounding. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 311–322, Tokyo, Japan. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

The ReproGen Shared Task on Reproducibility of Human Evaluations in NLG: Overview and Results

Anya Belz
ADAPT Research Centre, DCU, Ireland
anya.belz@adaptcentre.ie

Anastasia Shimorina
Orange, Lannion, France
anastasia.shimorina@orange.com

Shubham Agarwal
Heriot Watt University, UK
sa201@hw.ac.uk

Ehud Reiter
University of Aberdeen, UK
e.reiter@abdn.ac.uk

Abstract

The NLP field has recently seen a substantial increase in work related to reproducibility of results, and more generally in recognition of the importance of having shared definitions and practices relating to evaluation. Much of the work on reproducibility has so far focused on metric scores, with reproducibility of human evaluation results receiving far less attention. As part of a research programme designed to develop theory and practice of reproducibility assessment in NLP, we organised the first shared task on reproducibility of human evaluations, ReproGen 2021. This paper describes the shared task in detail, summarises results from each of the reproduction studies submitted, and provides further comparative analysis of the results. Out of nine initial team registrations, we received submissions from four teams. Meta-analysis of the four reproduction studies revealed varying degrees of reproducibility, and allowed very tentative first conclusions about what types of evaluation tend to have better reproducibility.

1 Introduction

There has been growing interest in reproducibility across Natural Language Processing (NLP) over recent years.[1] However, work has mostly focused on determining what information and resources need to be shared to enable others to obtain the same metric results. The reproducibility of human evaluation has received far less attention and currently very little is known about how reproducible, hence trustworthy, the human evaluations we routinely apply in NLP really are. This is of particular concern in Natural Language Generation (NLG) where human evaluations have always played a central role (Reiter, 2018; Novikova et al., 2017).

The last few years have seen a growth in publications, projects, workshops, shared tasks and other initiatives on the topic of reproducibility. For example, NeurIPS'19 introduced the ML Reproducibility checklist for submitted papers (Pineau et al., 2020) which was also adopted by EMNLP'20 and AAAI'21. The Reproducibility Challenge has been running since 2018, initially in conjunction with ICLR then NeurIPS (Sinha et al., 2020). The Challenge is focused on ML results and metric scores, and is organised as a 'live' challenge, where participants pick one of the accepted papers, and try to reproduce its ML results (Sinha et al., 2020).

The REPROLANG'20 shared task (Branco et al., 2020) asked participants to reproduce results from 11 papers in different areas of NLP. While in the case of ten of the papers, the results up for reproduction were automatic scores, in one case (Nisioi et al., 2017) they included human evaluation scores.[2] In their reproduction study of this work, Cooper and Shardlow (2020) reannotated original system outputs using their own annotators, in order to be able to compare annotation results. Their results suggested a drop in both quality metrics of close to 15%.

Apart from the above reproduction study involving text simplification carried out within REPROLANG, there appears to have been just one other paper reporting reproduction studies of human evaluation results in NLG (Belz and Kow, 2011) which re-ran two evaluation experiments

[1] We carried out a systematic review of reproducibility research in NLP in part as background research for ReproGen (Belz et al., 2021).

[2] Task D.1: Text simplification: http://wordpress.let.vupr.nl/lrec-reproduction/

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 249–258,
Aberdeen, Scotland, UK, 20-24 September 2021.

with the same evaluator cohorts, one in data-to-text generation, the other in visual referring expression generation. Here, strong correlations between annotator scores were found for two quality criteria for each task, 0.87 Pearson's in one case, >0.94 in the other three.

With the ReproGen shared task, our aim was to add to this currently small body of literature, in order to shed more light on how reproducible current human evaluation methods are, and what we may need to change in how we design and carry out human evaluations in order to improve reproducibility. In Section 2 we start by describing the organisation and structure of the shared task. Next we provide an overview of the participating teams (Section 3) and look at the properties of submitted systems (Section 4). We compare and analyse the results from the submitted systems in detail (Section 5), before we conclude with some discussion (Section 6) and tentative conclusions (Section 7).

2 Organisation of Shared Task

ReproGen'21[3] had two tracks, one a shared task in which teams try to reproduce the same prior human evaluation results, the other an 'unshared task' in which teams attempt to reproduce their own prior human evaluation results:

A ***Main Reproducibility Track:*** For a shared set of selected human evaluation studies, participants repeat one or more studies, and attempt to reproduce the results, using published information plus additional information and resources provided by the authors, and making common-sense assumptions where information is still incomplete.

B ***RYO Track:*** *R*eproduce *Y*our *O*wn previous human evaluation results, and report what happened. Unshared task.

For the main track (A above), we issued a call for proposals of papers, asking people to propose papers via an online form.[4] This yielded seven proposed papers, from which we selected four on the grounds of suitability for reproduction studies, diversity of languages and cost of reproduction. The selected papers and studies, with many thanks to the authors for supporting ReproGen, are:

1. van der Lee et al. (2017): *PASS: A Dutch data-to-text system for soccer, targeted towards specific audiences*: 1 evaluation study; Dutch; 20 evaluators; 3 quality criteria; reproduction target: primary scores.
2. Dušek et al. (2018): *Findings of the E2E NLG Challenge*: 1 evaluation study; English; MTurk; 2 quality criteria; reproduction target: primary scores.
3. Qader et al. (2018): *Generation of Company descriptions using concept-to-text and text-to-text deep models: dataset collection and systems evaluation*: 1 evaluation study; English; 19 evaluators; 4 quality criteria; reproduction target: primary scores.
4. Santhanam and Shaikh (2019): *Towards Best Experiment Design for Evaluating Dialogue System Output*: 3 evaluation studies differing in experimental design; English; 40 evaluators; 2 quality criteria; reproduction target: correlation scores between 3 studies.

Authors of original papers in Track A were asked (i) to complete a HEDS datasheet[5] (Shimorina and Belz, 2021) for their paper, (ii) to make available all code and other resources needed for the study, and (iii) to be available to answer questions and provide other help during the ReproGen participation period. Authors of reproduction papers were also asked to complete a HEDS datasheet.

We issued a call for participation, inviting teams to participate in one or both tracks. Nine teams registered for ReproGen, with team members from five different countries, out of which four teams submitted reproduction studies. Details of the submitting teams can be found in the following section.

We made available broad guidelines[6] to participating teams about how to report reproduction results, and provided light-touch review with comments and feedback on papers.

3 Overview of Participants and Submissions

Four submissions were received by the deadline on August 15, 2021. Two of the submissions were from Germany, one from Ireland, and one was a collaboration between groups in Spain, Brazil and Ireland. Two of the teams participated in Track A

[3] All information and resources relating to ReproGen are available at https://reprogen.github.io/

[4] https://forms.gle/J5ranvXqmfjPDbxLA

[5] https://forms.gle/MgWiKVu7i5UHeMNQ9

[6] https://reprogen.github.io/submissions/

Track	Team	Original paper	Reproduction paper
A	Technical University of Darmstadt (TUDA)	Qader et al. (2018)	Richter et al. (2021)
	UPF Barcelona, UF Minas Gerais, ADAPT Dublin	van der Lee et al. (2017)	Mille et al. (2021)
B	Trivago GmbH, Düsseldorf	Mahamood et al. (2007)	Mahamood (2021)
	ADAPT Dublin	Popović (2020)	Popović and Belz (2021)

Table 1: Overview of ReproGen submissions (tracks, teams, original papers and reproduction reports).

(Mille et al., 2021; Richter et al., 2021), the other two in Track B (Mahamood, 2021; Popović and Belz, 2021). Three of the four teams are affiliated with universities, one with a commercial company.

Each of the submissions reported a reproduction study for a different paper. Two of the evaluated systems produced outputs in English, one in Croatian, and one in Dutch. While Mahamood (2021) and Mille et al. (2021) reproduced human evaluation of data-to-text systems, Popović and Belz (2021) evaluated Machine Translation (MT) systems and Richter et al. (2021) text-to-text and concept-to-text generation systems. An overview of all submissions is provided in Table 1, and the properties of participating systems and studies are discussed in more detail in the next section.

4 Comparison of Properties of Original vs. Reproduction Studies

Overall, all teams tried to follow the original studies as closely as possible. All of the reproduction studies evaluated the same texts as reported in the original experiments, with the same criteria and measurement methods. Three of the four submissions used the same number of evaluators. Cohorts of human evaluators involved were different across all pairs of original and reproduction studies.

Below we summarise differences in each pair of studies and highlight the possible factors that might have affected reproduction results. In the case of Track A contributions, our notes are based on the HEDS datasheets completed by both the original study authors and the shared task participants. For Track B, we describe differences as reported by the authors themselves in their original and reproduction reports, also consulting the HEDS sheets completed by them.

See also Table 3 which lists some of the more fine-grained information for each study from the HEDS sheets.

4.1 Track A

Mille et al. (2021) reproduced van der Lee et al. (2017), the main differences being recruitment process and means of response collection. The original study recruited people on campus where they filled paper forms in one sitting, whereas the reproduction study used online surveys, where there was no control for timing, and people were recruited via personal contacts, i.e. they also included people known to the authors. The online form the authors used was designed to resemble the original paper form as much as possible. In addition, the reproduction study carried out some quality checks after the survey completion and replaced one entry from one participant, while the original experiment did not have any quality assurance methods (and consequently had some missing values).

Richter et al. (2021) reproduced Qader et al. (2018). Similar to the previous reproduction study of Mille et al. (2021), the main differences lie in survey design, and participant recruitment and background. While the original study used a specific web-based interface, the reproduction study built a Google form. That led to some differences in the interface, e.g. using checkboxes instead of a slider in the original evaluation. As regards human participants, the original evaluation was circulated among the authors' colleagues in their research lab; in contrast, the reproduction was carried out with friends and acquaintances. Both studies assessed English text in non-English-speaking countries; there was no formal assessment of the level of English among participants. Finally, manual quality checking was present in both studies after the evaluation experiment (for details, see the two papers); this involved subjective judgements and is hard to repeat across two studies.

4.2 Track B

Mahamood (2021) reproduced Mahamood et al. (2007). The original study used paper forms, while the reproduction used an online form. Evaluators were Master students in the original; the reproduction study instead used work colleagues. Another difference consists in the number of evaluators involved. There were 25 participants in the part of the original study that was reproduced; in contrast, the

Measurand(s)	Pearson's r	Spearman's ρ	mean % change		mean CV*
			+/-	abs	
Original study = van der Lee et al. (2017); reproduction study = Mille et al. (2021):					
All scores (1 system × 3 measures)	0.999**	1	10.19	10.19	11.891
Original study = Mahamood et al. (2007); reproduction study = Mahamood (2021):					
All scores (2 scenarios × 2 evaluator cohorts)	0.085	0.4	-24.14	60.16	72.343
Original study = Popović (2020); reproduction study = Popović and Belz (2021):					
Comprehension Minor, % words with errors (3 systems)	0.666	0.993	26.033	26.033	22.143
Comprehension Major, % words with errors (3 systems)	0.988*	0.973	47.953	47.953	38.227
Adequacy Minor, % words with errors (3 systems)	0.362	0.277	0.350	17.210	17.830
Adequacy Major, % words with errors (3 systems)	0.9986**	0.9997	48.443	48.443	38.667
All Scores (3 systems × 4 measures)	0.691**	0.818**	30.695	34.910	29.217
Original study = Qader et al. (2018); reproduction study = Richter et al. (2021):					
Mean Information Coverage (7 systems)	0.567	0.3397	36.826	42.840	34.044
Mean Non-redundancy of Information (7 systems)	0.328	0.524	1.899	19.153	19.108
Mean Semantic Adequacy (7 systems)	0.514	0.378	-2.979	19.201	20.396
Mean Grammatical Correctness (7 systems)	0.322	0.136	4.600	16.003	15.089
All Scores (7 systems × 4 measures)	0.679**	0.343	10.086	24.299	22.159

Table 2: Pearson's and Spearman's correlation coefficients, mean percentage change, and mean coefficients of variation (CV*), for the ReproGen'21 reproduction studies. ∗∗ = statistically significant at $\alpha = .01$, ∗ = at $\alpha = .05$.

reproduction study had 11 evaluators. Furthermore, the ratios between native and fluent English speakers were not the same: 14 and 11 in the original vs. 5 and 6 in the reproduction. Such distinctions may impact the reproduction results, since the experiment examines the effect of hedges on native versus fluent English speakers.

Popović and Belz (2021) carried out a reproduction study of Popović (2020). The reproduction study followed the original closely, with the main difference in participant background. While students and researchers in computational linguistics with different levels of MT experience took part in the original study, the reproduction study involved translation students with roughly the same level of MT experience.

5 Comparing Reproducibility in the ReproGen Studies

Table 4 shows results from all submissions, in terms of the individual pairs of scores reported in original and reproduction paper (columns 2 and 3), the percentage increase or decrease from original to reproduction score (column 4), and the de-biased coefficient of variation, CV* (last column), following Belz (2021). The coefficient of variation (CV) is a standard measure of precision used in metrological studies to quantify reproducibility of measurements. Unlike mean and standard deviation, CV is not in the unit of the measurements, and captures the amount of variation there is in a set of n scores in a general way, providing a quantification of precision (degree of reproducibility) that is comparable across studies (Ahmed, 1995, p. 57). Note that we have shifted all evaluation scales to start at zero, to ensure fair comparison across evaluations, because both percentage change and CV in general underestimate variation for scales with a lower end greater than 0. Rather than standard CV, we use CV*, a de-biased version of CV, Belz (2021), because sample size (number of repeat measures) tends to be very small in NLP.[7]

CV* in Table 4 ranges from 6.107 to 16.372 for Mille et al. (2021)'s reproduction study; from 52.806 to 101.894 for Mahamood (2021); from 4.86 to 47.17 for Popović and Belz (2021); and from 0 to 66.467 for Richter et al. (2021). Percentage change gives a similar picture, as the two measures generally give similar results for sample size 2 (Pearson's correlation for absolute percentage change and CV* is 0.89 over all scores in Table 4).

Looking at the above CV* ranges for each reproduction study, a first indication of a ranking emerges for the four study pairs in terms of degree of reproducibility, with (1) Lee et al./Mille et al. having the highest degree of reproducibility, followed by (2) Popović/Popović & Belz, (3) Qader et al./Richter et al., and finally (4) Mahamood et al./Mahamood.

Table 2 provides higher-level results, where in each case multiple score pairs are analysed jointly,

[7]For full details of, and rationale for, using CV*, even for sets of just two scores, see Belz (2021).

Studies/measurands	3.1.1	3.2.1	4.3.4	4.3.8	4.1.1	4.1.2	4.1.3	scores /item	(mean) CV*
Lee et al./Mille et al.									11.891
Stance ID Acc	10	20/20	stance A, stance B	output classif	Feature	Both	EFoR	20	6.107
Clarity S3 ('Understandability')	20	20/20	1–7	DQE	Good	Both	iiOR	20	12.031
Clarity S4 ('Clarity')	20	20/20	1–7	DQE	Good	Both	iiOR	20	14.605
Fluency S1 ('Grammaticality')	20	20/20	1–7	DQE	Corr	Form	iiOR	20	18.303
Fluency S2 ('Readability')	20	20/20	1–7	DQE	Good	Both	iiOR	20	13.711
Popović/Popović & Belz									29.217
Comprehension Minor	557,	7/7	2 labels	Anno	Good	Both	iiOR	2	22.143
Comprehension Major	279,	7/7	2 labels	Anno	Good	Both	iiOR	2	38.227
Adequacy Minor	467	7/7	3 labels	Anno	Corr	Cont	RtI	2	17.830
Adequacy Major		7/7	3 labels	Anno	Corr	Cont	RtI	2	38.667
Qader et al./Richter et al.									22.159
Information Coverage	30	19/19	1–5	DQE	Corr	Cont	RtI	1	34.044
Information Non-redundancy	30	19/19	1–5	DQE	Good	Cont	iiOR	1	19.108
Semantic Adequacy	30	19/19	1–5	DQE	Corr	Cont	iiOR	1	20.396
Grammatical Correctness	30	19/19	1–5	DQE	Corr	Form	iiOR	1	15.089
Mahamood et al./Mahamood, Binary Preference Strength	2†	25‡/11	-3..+3	RQE	Good	Both	EFoR	25/11	72.343

Table 3: Summary of some properties from HEDS datasheets provided by ReproGen participants. 3.1.1 = number of items assessed per system; 3.2.1 = number of evaluators in original/reproduction experiment; 4.3.4 = List/range of possible responses; 4.3.8 = Form of response elicitation (DQE: direct quality estimation, RQE: relative quality estimation, Anno: evaluation through annotation); 4.1.1 = Correctness/Goodness/Features; 4.1.2 = Form/Content/Both; 4.1.3 = each output assessed in its own right (iiOR) / relative to inputs (RtI) / relative to external reference (EFoR); scores/item = number of evaluators who evaluate each evaluation item; (mean) CV*. † considering texts with and without hedges to be the two systems being compared. ‡ subset of 32 evaluators from original studies: 14 native + 11 fluent speakers.

in terms of Pearson's and Spearman's correlation coefficients (columns 2 and 3), mean percentage change and mean *absolute* percentage change (columns 4 and 5), and mean CV* (last column). For example, for Lee et al./Mille et al., Pearson's r was 0.99 for the three scores in the original study compared with the corresponding scores from the reproduction study, both as shown in Table 4; Spearman's ρ was 1 (i.e. all ranks were the same); on average scores went up by 10.19%; the absolute percentage change was also 10.19% (because all changes were positive); and on average CV* was 11.89. Where a study compared multiple systems in absolute terms,[8] we show results per evaluation measure (e.g. Comprehension Minor), in addition to results for all scores.

In terms of the study-level scores ('All Scores' rows) in Table 2, a more mixed picture emerges compared to Table 4. In terms of both Pearson's and Spearman's, the ranking is the same in Table 2 and Table 4: (1) Lee et al./Mille et al., (2) Popović/Popović & Belz, (3) Qader et al./Richter et al., then (4) Mahamood et al./Mahamood. In contrast, the rankings for overall mean (absolute) percentage change and overall mean CV* are slightly different: (1) Lee et al./Mille et al., (2) Qader et al./Richter et al., (3) Popović/Popović & Belz, then (4) Mahamood et al./Mahamood.

In Table 3, we summarise some properties of our four pairs of studies, in terms of a subset of the properties from the HEDS datasheet (Shimorina and Belz, 2021) we asked participants to complete,[9] to attempt to identify possible relationships between study properties and degree of reproducibility. As discussed in the next section, such interpretations could be made with greater confidence if sample sizes were larger than 2, and we intend to add further studies in the future to enable more confident conclusions.

Something that's not easily captured in a table is the differences in cohorts of evaluators. For example, in Mahamood et al./Mahamood, evaluators in the original study were students, whereas non-students were used in the reproduction study; the former were a lot younger on average. In Lee et al./Mille et al., the original study used random people encountered in the university's science building, the reproduction study used present and

[8] Mahamood et al./Mahamood assess two systems, but in relative terms, yielding just one score.

[9] We corrected the information provided in a small number of cases by referring to the papers.

former staff and postgraduate students in computing science some of whom were known to the authors; here too the former were a lot younger on average. In Popović/Popović & Belz, the original evaluators were computational linguistics staff and students, the evaluators in the reproduction study were translation students. Finally, in Qader et al./Richter et al., the original evaluators were recruited from among people in the same lab (excluding the authors), whereas the reproduction study authors recruited people from their social environment. Broadly speaking, differences between evaluator cohorts would seem to be particularly pronounced in Qader et al./Richter et al. and Mahamood et al./Mahamood, and these two study pairs are also the least reproducible out of the four study pairs, according to all measures except *mean* absolute percentage change and *mean* CV*.

In Table 3, column 2 shows the number of items assessed per system (Question 3.1.1 in the HEDS datasheet); column 3 shows the number of evaluators in an experiment (Question 3.2.1 in HEDS); column 4 shows the list/range of possible responses (4.3.4); column 5 shows the form of response elicitation (4.3.8); column 6 shows whether the underlying quality criterion assesses the correctness, goodness, or a feature-type aspect of quality (4.1.1); column 7 shows whether the quality criterion assesses an output's form, content or both (4.1.2); column 8 shows whether the quality criterion assesses each output in its own right (iiOR), relative to input (RtI), or relative to an external frame of reference (EFoR) (4.1.3); column 9 ('scores/item') shows the number of scores collected per evaluation item; finally, the last column shows corresponding mean CV* values for ease of reference. For full details regarding HEDS questions and possible values, see Shimorina and Belz (2021).

In Lee et al./Mille et al., Clarity and Fluency are compound measures each derived from two separately assessed quality criteria, which map to the normalised quality criterion names shown in rows 4–7 in Table 3, following the taxonomy of normalised quality criteria proposed by Howcroft et al. (2020).

Looking at Table 3, it's hard to detect any specific patterns in study properties that might be predictive of CV*. There is perhaps some indication that the (normalised) Grammaticality criterion has similar, and good, reproducibility in the three studies that use it in some guise: CV*= 19.3 for S1 in Lee et al./Mille et al.; 17.8 for Adequacy Minor[10] in Popović/Popović & Belz; and 15.1 for Grammatical Correctness in Qader et al./Richter et al. Moreover, the study with the highest degree of reproducibility according to all measures (Lee et al./Mille et al.) obtained a comparatively large number of scores for each evaluated item, while also assessing a medium number of items per system. In contrast, the study with the lowest degree of reproducibility according to all measures (Mahamood et al./Mahamood) obtained a different number of scores for each evaluated item in the original and reproduction studies, while assessing a very small number (2) of items per system. We return to some of these aspects in the discussion section.

6 Discussion

There were considerable differences in evaluator cohorts between original and reproduction study in all four ReproGen study pairs. In Mahamood et al./Mahamood, the texts being evaluated were about progress towards getting a postgraduate degree (e.g.: *You haven't qualified for a postgraduate diploma. You have been awarded a postgraduate certificate instead. Average CAS results were achieved in CS5052, CS5038, CS5540, CS5548.*) Mahamood et al. (2007) asked postgraduate students to evaluate these texts, whereas Mahamood (2021) asked work colleagues to evaluate the texts. It is possible that students and non-students reacted differently to statements about degree progress, and that the students were much more familiar with terms such as 'postgraduate certificate' and 'CAS'.

There were also important differences in evaluator cohorts in Lee et al./Mille et al. and Qader et al./Richter et al.: in both cases, the reproduction cohort included people known to the authors personally who may have had more of an incentive to perform the task conscientiously and perhaps also to select higher scores. In the case of Lee et al./Mille et al., reproducibility was nevertheless good, whereas for Qader et al./Richter et al., it was less good.

Across all of our reproduction studies, there were differences in evaluators: age, recruitment, professional status, domain knowledge, background, etc. Such differences have the potential to impact reproducibility, but the picture from the four ReproGen studies was mixed, and further research is needed

[10] Assuming that grammatical errors account for much of minor translation adequacy issues, which is not certain.

to understand which characteristics were most important from this perspective. Knowing this would be very helpful in designing and interpreting experiments, as well as replicating them.

Both Track A reproduction studies contacted the original authors for additional information, highlighting the importance of original authors being willing to support reproduction studies of their work. It is clear from ReproGen'21 as well as other research (van der Lee et al., 2019; Howcroft et al., 2020; Belz et al., 2021) that we need a lot of information about evaluators and other aspects of evaluations in order to conduct reproduction studies, so it's essential that experimenters fill out a datasheet such as HEDS which conveys information in a standardised, comparable way.

A rarely mentioned aspect that should not be underestimated is that being willing to support a reproduction study of your work means being willing to take what some perceive as a substantial risk associated with having others publish assessments of the reproducibility of your work. Some authors are very uncomfortable with a reproduction study showing low reproducibility. In fact, one of the authors of a paper which had been the subject of a reproduction study that we wanted to include in our survey of reproduction studies (Belz et al., 2021) worried that the considerable gap in results would be interpreted as academic misconduct.[11]

Clearly there is a need for reproduction studies to be carried out in NLP. We need to know how reproducible different types of evaluation measures are, because measures with low reproducibility will result in unreliable results and unreliable conclusions based on them. Reproduction studies are the only way to know if/where we're going wrong in this sense. However, given prevailing sensitivities, it seems the right thing to do to conduct reproduction studies with the original authors' consent.

Reproduction studies are expensive and a lot work, and we were told by the five teams that registered for ReproGen but did not submit that these were the main reasons why they were ultimately not able to participate. Publication only provides so much of an incentive/motivation. Significant numbers of reproduction studies may only be feasible in the context of a funded project such as ReproHum,[13] where uniformity of approach can moreover be ensured and the number and type of reproduction studies conducted can be more directly controlled. We plan to run a second shared task next year, to further test the suitability of the shared task format for reproduction studies in NLP.

7 Conclusions

We first proposed the ReproGen shared task at Generation Challenges 2020[12] (Belz et al., 2020) and, taking into account feedback received, developed it into the shared task presented here, with the main track offering four original studies (sets of human evaluation results) for reproduction, and an open track inviting reproduction studies of own results.

Bearing in mind we had just one reproduction study for each original study available to us, and that as discussed we have to be cautious drawing conclusions based on sample sizes of 2, there are very few tentative first conclusions concerning reproducibility of human evaluation in NLG we have been willing to draw from ReproGen. We pointed out that the study with the highest degree of reproducibility obtained a comparatively large number of scores for each evaluated item, while also assessing a medium number of items per system. In contrast, the study with the lowest degree of reproducibility obtained a different number of scores for each evaluated item in the original and reproduction studies, while assessing a very small number (2) of items per system. We also observed that there was some evidence that the Grammaticality evaluation criterion has a comparatively good and stable degree of reproducibility.

When we read human evaluation results in NLG papers, unless there is an obvious red flag such as a very small number of evaluators, or evaluation items, we tend to trust those results more than metric results. Yet as we delve deeper into the reproducibility of our human evaluation results, it is beginning to become clear that, as a general assumption, this trust may be misplaced. More generally, that we need to do much more as a field to ensure that our human evaluation methods are fit for purpose, including in the sense that a rerun of an experiment will produce at least broadly similar results. With the ReproGen shared task, and the ReproHum project[13] which it is part of, we are aiming to make a contribution to this important goal.

[11] We therefore did not include the study in question in the published survey.

[12] INLG'20, Dublin.

[13] https://gow.epsrc.ukri.org/NGBOViewGrant.aspx?GrantRef=EP/V05645X/1

Acknowledgments

We thank the authors of the four original papers that were up for reproduction in Track A who bravely agreed to be our guinea pigs for this first shared task on reproducibility of evaluation measures in NLG. And of course the authors of the reproduction papers, the first batch of participants in a shared task on reproducibility of human evaluations without whom there would be no shared task.

Our work was carried out as part of the ReproHum project on Investigating Reproducibility of Human Evaluations in Natural Language Processing, funded by EPSRC (UK) under grant number EP/V05645X/1.

Shubham Agarwal's PhD fees are supported by Adeptmind Inc., Toronto, Canada.

References

S. E. Ahmed. 1995. A pooling methodology for coefficient of variation. *Sankhyā: The Indian Journal of Statistics, Series B*, pages 57–75.

Anya Belz. 2021. Quantifying reproducibility in NLP and ML. *arXiv preprint arXiv:2109.01211*.

Anya Belz, Shubham Agarwal, Anastasia Shimorina, and Ehud Reiter. 2020. ReproGen: Proposal for a shared task on reproducibility of human evaluations in NLG. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 232–236.

Anya Belz, Shubham Agarwal, Anastasia Shimorina, and Ehud Reiter. 2021. A systematic review of reproducibility research in natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 381–393, Online. Association for Computational Linguistics.

Anya Belz and Eric Kow. 2011. Discrete vs. continuous rating scales for language evaluation in NLP. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 230–235.

António Branco, Nicoletta Calzolari, Piek Vossen, Gertjan Van Noord, Dieter van Uytvanck, João Silva, Luís Gomes, André Moreira, and Willem Elbers. 2020. A shared task of a new, collaborative type to foster reproducibility: A first exercise in the area of language science and technology with REPROLANG2020. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5539–5545, Marseille, France. European Language Resources Association.

Michael Cooper and Matthew Shardlow. 2020. CombiNMT: An exploration into neural text simplification models. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5588–5594, Marseille, France. European Language Resources Association.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands. Association for Computational Linguistics.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368, Tokyo, Japan. Association for Computational Linguistics.

Chris van der Lee, Emiel Krahmer, and Sander Wubben. 2017. PASS: A Dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 95–104, Santiago de Compostela, Spain. Association for Computational Linguistics.

Saad Mahamood. 2021. Reproducing a comparison of hedged and non-hedged NLG texts. In *Proceedings of the 14th International Conference on Natural Language Generation*, Aberdeen, United Kingdom. Association for Computational Linguistics.

Saad Mahamood, Ehud Reiter, and Chris Mellish. 2007. A comparison of hedged and non-hedged nlg texts. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*, pages 155–158.

Simon Mille, Thiago Castro Ferreira, Anya Belz, and Brian Davis. 2021. Another PASS - a reproduction study of the human evaluation of a football report generation system. In *Proceedings of the 14th International Conference on Natural Language Generation*, Aberdeen, United Kingdom. Association for Computational Linguistics.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 2: Short Papers), pages 85–91, Vancouver, Canada. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. 2020. Improving reproducibility in machine learning research (a report from the NeurIPS 2019 reproducibility program). *CoRR abs/2003.12206*.

Maja Popović. 2020. Informative manual evaluation of machine translation output. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5059–5069, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Maja Popović and Anya Belz. 2021. A reproduction study of an annotation-based human evaluation of MT outputs. In *Proceedings of the 14th International Conference on Natural Language Generation*, Aberdeen, United Kingdom. Association for Computational Linguistics.

Raheel Qader, Khoder Jneid, François Portet, and Cyril Labbé. 2018. Generation of company descriptions using concept-to-text and text-to-text deep models: dataset collection and systems evaluation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 254–263, Tilburg University, The Netherlands. Association for Computational Linguistics.

Ehud Reiter. 2018. A structured review of the validity of BLEU. *Computational Linguistics*, 44(3):393–401.

Christian Richter, Yanran Chen, and Steffen Eger. 2021. TUDA-reproducibility @ reprogen: Replicability of human evaluation of text-to-text and concept-to-text generation. In *Proceedings of the 14th International Conference on Natural Language Generation*, Aberdeen, United Kingdom. Association for Computational Linguistics.

Sashank Santhanam and Samira Shaikh. 2019. Towards best experiment design for evaluating dialogue system output. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 88–94, Tokyo, Japan. Association for Computational Linguistics.

Anastasia Shimorina and Anya Belz. 2021. The human evaluation datasheet 1.0: A template for recording details of human evaluation experiments in NLP. *CoRR*, abs/2103.09710.

Koustuv Sinha, Joelle Pineau, Jessica Forde, Rosemary Nan Ke, and Hugo Larochelle. 2020. NeurIPS 2019 Reproducibility Challenge. *ReScience C*, 6(2):#11.

Measurand	Orig study	Repro study	% change	CV*
Original study = van der Lee et al. (2017); reproduction study = Mille et al. (2021):				
Stance identification Accuracy, PASS system	91	96.75	6.32	6.107
Mean Clarity, 0..6†, PASS system	4.64	5.3	10.7	13.193
Mean Fluency, 0..6†, PASS system	4.36	5.14	13.55	16.372
Original study = Mahamood et al. (2007); reproduction study = Mahamood (2021):				
Strength of preference for style A vs. B (0..6†)				
Native speakers, Scenario 1	1.58	0.8	-49.37	65.35
Native speakers, Scenario 2	0.93	1.6	72.04	52.806
Fluent speakers, Scenario 1	3.09	1.0	-67.64	101.894
Fluent speakers, Scenario 2	3.45	1.67	-51.59	69.323
Original study = Popović (2020); reproduction study = Popović and Belz (2021):				
Comprehension Minor, % words with errors				
Bing	16.0	16.8	+5	4.86
Google	11.2	15.0	+33.93	28.92
Amazon	12.0	16.7	+39.17	32.65
Comprehension Major, % words with errors				
Amazon	7.6	10.2	+34.21	29.13
Bing	15.1	22.3	+47.68	38.38
Google	7.1	11.5	+61.97	47.17
Adquacy Minor, % words with errors				
Google	10.5	11.7	+11.43	10.78
Amazon	11.4	13.1	+14.91	13.84
Bing	17.0	12.7	-25.29	28.87
Adequacy Major, % words with errors				
Google	7.0	9.7	+38.57	32.24
Amazon	6.5	9.5	+46.15	37.39
Bing	13.2	21.2	+60.61	46.37
Original study = Qader et al. (2018); reproduction study = Richter et al. (2021):				
Mean Information Coverage, 0..4†				
Reference	2.1	2.9	38.1	31.904
C2T	1.9	1.5	-21.05	23.459
C2T_char	1.3	2.0	53.85	42.297
C2T+pg	1.3	1.6	23.08	20.628
C2T+pg+cv	1.7	2.0	17.65	16.168
T2T+pg	0.8	1.6	100	66.467
T2T+pg+cv	1.3	1.9	46.15	37.388
Mean Non-redundancy of Information, 0..4†				
Reference	3.6	3.1	-13.89	14.881
C2T	1.9	2.8	47.37	38.183
C2T_char	2.9	1.8	-37.93	46.668
C2T+pg	3.5	3.2	-8.57	8.928
C2T+pg+cv	2.9	3.1	6.9	6.647
T2T+pg	2.3	2.5	8.7	8.308
T2T+pg+cv	2.8	3.1	10.71	10.139
Mean Semantic Adequacy, 0..4†				
Reference	2.9	2.9	0	0
C2T	2.3	1.6	-30.43	35.79
C2T_char	1.8	2.1	16.67	15.339
C2T+pg	3.0	1.9	-36.67	44.764
C2T+pg+cv	2.6	2.9	11.54	10.876
T2T+pg	1.9	1.7	-10.53	11.078
T2T+pg+cv	1.4	1.8	28.57	24.925
Mean Grammatical Correctness, 0..4†				
Reference	3.2	3.0	-6.25	6.432
C2T	2.6	2.2	-15.38	16.617
C2T_char	2.0	2.5	25	22.156
C2T+pg	3.3	2.8	-15.15	16.344
C2T+pg+cv	3.2	3.1	-3.13	3.165
T2T+pg	2.7	3.0	11.11	10.495
T2T+pg+cv	2.5	3.4	36	30.417

Table 4: Overview of results from ReproGen'21 reproduction studies: measurand, measured value in original study, measured value in reproduction study, percentage change (in/decrease), and coefficient of variation (CV*). † the original scale was shifted to start from 0.

Text-in-Context: Token-Level Error Detection for Table-to-Text Generation

Zdeněk Kasner,[1] **Simon Mille**[2] and **Ondřej Dušek**[1]
[1]Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic
[2]Pompeu Fabra University, Barcelona, Spain
kasner@ufal.mff.cuni.cz, simon.mille@upf.edu, odusek@ufal.mff.cuni.cz

Abstract

We present our Charles-UPF submission for the Shared Task on Evaluating Accuracy in Generated Texts at INLG 2021. Our system can detect the errors automatically using a combination of a rule-based natural language generation (NLG) system and pretrained language models (LMs). We first utilize a rule-based NLG system to generate sentences with facts that can be derived from the input. For each sentence we evaluate, we select a subset of facts which are relevant by measuring semantic similarity to the sentence in question. Finally, we finetune a pretrained language model on annotated data along with the relevant facts for fine-grained error detection. On the test set, we achieve 69% recall and 75% precision with a model trained on a mixture of human-annotated and synthetic data.

1 Introduction

Recent neural NLG systems can easily generate fluent texts from linearized structured data (Zhao et al., 2020; Kale and Rastogi, 2020; Castro Ferreira et al., 2020). However, the systems cannot guarantee that the output is properly grounded in the input – hallucination (outputs not supported by input data) is a notorious problem in neural NLG (Tian et al., 2019; Harkous et al., 2020; Filippova, 2020; Rebuffel et al., 2021). Neural systems are particularly unreliable on complex datasets such as Rotowire (Wiseman et al., 2017), where the task is to generate basketball match summaries from tabular data. Rotowire poses multiple challenges for neural systems: it requires content selection and production of longer texts, and its human-written training texts are themselves not always grounded in data, which makes neural models more susceptible to hallucination.

On the other hand, rule-based systems used in recent data-to-text tasks (Lapalme, 2020; Tran and Nguyen, 2020; Mille et al., 2019) all achieve high scores in terms of accuracy of the generated contents with respect to the input structures (Dušek et al., 2020; Castro Ferreira et al., 2020). This, however, comes with the cost of lower fluency.

Detecting NLG errors automatically is a hard problem. For word-overlap-based metrics, such as BLEU (Papineni et al., 2002) or METEOR (Lavie and Agarwal, 2007), reliability on content checking is known to be poor (Novikova et al., 2017; Dhingra et al., 2019). Most neural metrics (Zhang et al., 2020; Sellam et al., 2020) have not been evaluated for content preservation. Dušek and Kasner (2020)'s metric based on natural language inference (NLI) specifically targets content preservation, but, same as all previously mentioned ones, is not able to provide fine-grained error tagging beyond sentence level. Specific content-checking metrics mostly remain a domain of handcrafted pattern matching (Wen et al., 2015; Dušek et al., 2019), which does not scale well to new domains. While human evaluation provides a more reliable alternative, it is costly and difficult to set up (van der Lee et al., 2019; Santhanam and Shaikh, 2019; Belz et al., 2020; Thomson and Reiter, 2020a).

The INLG 2021 accuracy evaluation shared task (Reiter and Thomson, 2020; Thomson and Reiter, 2021) aims to improve this situation. Reiter and Thomson (2020) carefully annotated 60 outputs of various neural systems trained on Rotowire with 6 error types (see Table 1) defined in Thomson and Reiter (2020b). The objective of the shared task is then to either implement an automatic metric for creating the same type of annotations automatically, or to develop a human evaluation scenario capable of producing the same annotations while requiring less resources.

Our submission for the shared task falls into the first category: we developed an automatic metric for token-level error annotation which combines a

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 259–265,
Aberdeen, Scotland, UK, 20-24 September 2021.

NUMBER	Incorrect number.
NAME	Incorrect named entity.
WORD	Any other incorrect word.
CONTEXT	A phrase inappropriate for the context.
NOT_CHECKABLE	A statement which cannot be checked.
OTHER	Any other type of mistake.

Table 1: Error categories for the Rotowire dataset.

rule-based generation system with a neural retrieval model and a pretrained neural LM used for error tagging. We evaluated our approach in a cross-validation scenario to select the best configuration for the shared task. Overall, our system is able to reach 65% error detection F1 score and ranked first out of four automatic submissions in the shared task. The code for our experiments is freely available on Github.[1]

2 Our System

Our system is composed of 3 steps: A rule-based generator for fact descriptions (see Figure 1 and Section 2.1), a retrieval system for selecting facts relevant for a given sentence (Section 2.2), and a token-level error tagger based on the RoBERTa pretrained LM (Section 2.3). The latter two steps are summarized in Figure 2. The LM tagger is trained on examples provided by shared task organizers, as well as on synthetic data based on the Rotowire training set (Section 2.4).

2.1 Rule-based Fact Descriptions

We use rule-based systems to generate natural language descriptions of facts from the input tables, relating to all players and both teams. The facts are later supplied to the error-checking model for grounding the evaluated sentence (see Section 2.3). We experiment with both *simple* descriptions created by filling in sentence templates, and *compact* descriptions generated using a grammar-based system. The simple system produces about 569 facts/sentences for each game. The compact system generates about 112 sentences per game, i.e., 5 times less; the game descriptions contain the same amount of information but the individual sentences are more syntactically complex.

Facts generated For each game, we first generate every fact in the input table, i.e., 44 facts about the game (hosting team, visiting team, date converted to weekday) and so-called line-score objects (team name and statistics) and box-score objects (player name, player team, player starting position and their personal statistics).

Subsequently, we generate 85 further facts that can be inferred from the input table. These are based on reading the first 20 human-written summaries in the training data and finding frequently mentioned facts that can easily be derived from input, such as which team won and by how much, comparisons between the team and player raw data (e.g., *Team A dominated the defensive rebounding*, *Team A and Team B committed the same number of fouls*; *Player X was the (second) best scorer of the game/his team*), complex statistics (e.g., *Team A totaled X steals*; *Player X (almost) recorded a double-double*), or an interpretation of some numbers (e.g., *Team A came back in the 4th quarter; Team A was efficient/bad at shooting*).[2]

Simple descriptions are produced by a template-based system, with one template per fact. We handcrafted 129 sentence templates to cover all the facts described above. A sentence template looks like the following: "*[PLAYER_NAME] scored [PTS] points.*", where square brackets indicate variables that are instantiated with the corresponding input values (see Figure 1 for sample sentences).

Compact descriptions are produced by the FORGe system (Mille et al., 2019), which allows for the generation of more compact sentences by instantiating abstract (predicate-argument) templates instead of full sentences for each fact. For instance, the template for the points scored would be: *[PLAYER_NAME] ←A1 provide A2→ point NonCore→[PTS]*, where A1 and A2 denote the first and second arguments respectively, and NonCore a non-argumental relation. FORGe receives a series of instantiated templates and performs surface realization in several steps, by first aggregating the templates based on predicate and/or argument identity, and then structuring, linearizing and inflecting components of the sentences. The FORGe grammars were used off-the-shelf,[3] with cross-sentence referring expression generation deactivated so that each generated sentence can be used on its own. We manually crafted 98 abstract templates and added a description of the included

[1] https://github.com/kasnerz/accuracySharedTask_CUNI-UPF

[2] A number of mentioned facts could not be obtained from the Rotowire data, as for instance the player stats per quarter, a career high points total, whether a player is an all-star or not, or if a player scored the winning shot.

[3] Minor debugging was needed to cover some new contexts.

Figure 1: Rule-based NLG which we use to generate facts from the input data. The facts are used as an input to the error checking model (see Figure 2). We experiment with (a) simple hand-crafted templates and (b) compact sentences generated by the FORGe system.

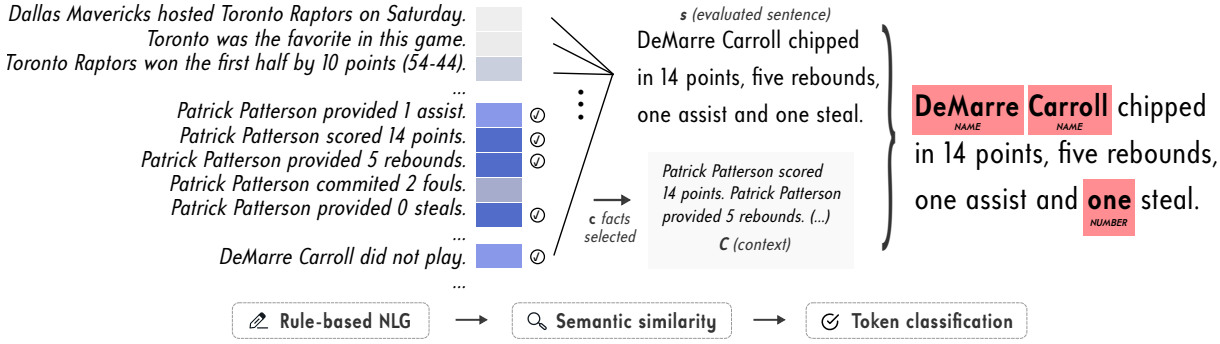


Figure 2: An overview of our system. First, we generate the facts from the input table with a rule-based NLG system (see Figure 1). For each evaluated sentence s, we select c facts with the highest semantic similarity, getting a context C. The pair (C, s) is given as an input to a pretrained LM for token-level error classification.

lexical units into the FORGe lexicon. For instance, the five simple sentences shown at the bottom of the yellow column in Figure 1 are covered by a single compact sentence shown at the bottom of the orange column.

2.2 Context Retrieval

Since the maximum length of the input sequence for our error-checking model (see Section 2.3) is 512 tokens (about 10% of the total length of the generated sentences G), we need to select only a subset of G, which we refer to as **context** C. We want to put the relevant sentences into C and filter out the rest to make the error tagging easier. This problem is hard in general, as any string matching (e.g. using numbers or names mentioned in the sentence) will fail on lexical variations.

Our solution is based on selecting sentences with the highest **semantic similarity**. For each generated sentence $g_i \in G$, we measure semantic similarity between g_i and the evaluated sentence s using Sentence Transformers (Reimers and Gurevych, 2019).[4] In particular, we embed the sentence tokens by applying mean pooling on the output of `paraphrase-distilroberta-base-v2`, getting the embedding vectors e_s and e_{g_i}. Then we compute the cosine similarity between the embeddings. For the context C, we select the top c sentences from G that have the highest cosine similarity to s.

2.3 LM-based Error Tagger

We use a RoBERTa LM (Liu et al., 2019) with a token-level classification head as our error-checking model. Unlike unsupervised approaches based on examining attention values (Thorne et al., 2019; Li et al., 2020) or input perturbations (Kim et al., 2020), we train the model directly to predict error categories using annotated data, similarly to Yoosuf and Yang (2019).

The model receives an input $X = (C, s)$, composed of the context C, i.e., relevant background facts selected by context retrieval in Section 2.2, and the generated sentence s to be tagged. The inputs are separated by the delimiter `</s>`. The model is trained to annotate each token in s either with an *OK* label, or with a label corresponding to one of the error categories.

We experiment with two data sources for training the model: (1) **gold-standard annotated data**

[4] https://www.sbert.net/

Generator	Data	c	EMR = 0.25			EMR = 0.5			EMR = 0.75		
			R	P	F1	R	P	F1	R	P	F1
Simple	synth	5	0.123	0.723	0.210	0.165	0.512	0.250	0.310	0.323	0.316
		10	0.138	0.737	0.232	0.181	0.549	0.272	0.328	0.400	0.360
		20	0.137	**0.741**	0.231	0.179	0.559	0.271	0.327	0.433	0.373
		40	0.165	0.712	0.268	0.199	0.560	0.294	0.296	0.436	0.353
	synth + human	5	0.422	0.617	0.501	0.414	0.594	0.488	0.401	0.583	0.475
		10	0.467	0.551	0.506	0.438	0.638	0.519	0.428	0.665	0.521
		20	0.518	0.640	0.573	0.544	0.575	0.559	0.509	0.595	0.549
		40	0.584	0.644	**0.613**	**0.595**	0.612	0.603	0.519	0.639	0.573
Compact	synth	5	0.151	**0.696**	0.248	0.170	0.617	0.267	0.336	0.427	0.376
		10	0.176	0.663	0.278	0.195	0.624	0.297	0.295	0.486	0.367
		20	0.196	0.672	0.303	0.205	0.635	0.310	0.278	0.552	0.370
		40	0.166	0.643	0.264	0.197	0.595	0.296	0.306	0.530	0.388
	synth + human	5	0.600	0.641	0.620	0.552	0.635	0.591	0.588	0.600	0.594
		10	0.583	0.662	0.620	0.629	0.606	0.617	**0.656**	0.606	0.630
		20	0.622	0.647	0.634	0.597	0.688	0.639	0.600	0.660	0.629
		40	0.614	0.690	**0.650***	0.609	0.630	0.619	0.611	0.630	0.620

Table 2: Recall (R), precision (P) and F1 scores on development data. c indicates the number of sentences in the context provided to the tagger, EMR stands for entity modification rate. Best recall, precision and F1 scores for both generators (simple and compact) are shown in bold, the submitted model is identified by an asterisk (*).

from the shared task (which contains all error types), (2) **synthetic data** created by perturbing the human-written summaries from Rotowire (which contains only *NAME* and *NUMBER* errors; see Section 2.4 for details).

2.4 Synthetic Data

The gold-standard data contains only 60 games, as opposed to 3,395 games in the Rotowire training set. This led us to an idea of using the training set as a source of synthetic data for our model.

We create the synthetic data by introducing errors into human-written descriptions. We focus only on the *NAME* and *NUMBER* errors—the categories which are the most represented and also easiest to generate. In each sentence, we identify named entities in the text using *spaCy*.[5] We modify only certain portion of entities according to the *entity modification rate*, which we treat as a hyperparameter. We introduce the *NAME* errors by:

(1) swapping the names of teams with opponent teams,
(2) swapping the names of players with other players in the game,
(3) swapping the names of cities with other cities in the Rotowire dataset,
(4) modifying the days of the week.

For *NUMBER* errors, we take an integer n identified in the text, sample a number from a normal distribution with $\mu = n$ and $\sigma = 3$, and truncate it to get the integer. We re-sample if the output equals the original number, or for negative outputs. If the number is spelled out, we use `text2num`[6] and `num2words`[7] to convert to digits and back.

3 Experiments

We train a PyTorch version of RoBERTa from the Huggingface Transfomers repository (Wolf et al., 2019) using the AdamW optimizer (Loshchilov and Hutter, 2019), learning rate 5×10^{-5} and linear warmup. We finetune the model for 10 epochs and select the model with the highest validation score. We experiment with several hyperparameters:

(a) *simple* vs. *compact* sentences in G,
(b) *number of sentences* retrieved for the context: c = 5, 10, 20 or 40;
(c) *entity modification rate* (EMR): proportion of entities which are modified in the synthetic data: 0.25, 0.5, or 0.75.

We evaluate the model using a script provided by the organizers, which computes recall and precision of the model output with respect to the human-annotated data. Since we use the human-annotated data for training, we perform 6-fold cross-validation: in each run, we use 45 games for training, 5 games for validation, and 10 games for evaluation.

The results of our model on the development data are listed in Table 2.[8] For our final submission,

[5] https://spacy.io

[6] https://pypi.org/project/text2num/

[7] https://pypi.org/project/num2words/

[8] Due to space constraints, we do not list the results of

Error Type	Mistake		Token	
	R	P	R	P
NAME	0.750	0.846	0.759	0.862
NUMBER	0.777	0.750	0.759	0.752
WORD	0.514	0.483	0.465	0.529
CONTEXT	0.000	-	0.000	-
NOT_CHECKABLE	0.000	-	0.000	-
OTHER	0.000	-	0.000	-
Overall	0.691	0.756	0.550	0.769

Table 3: Results of our system on test data: recall (R) and precision (P) are shown for individual error types.

we selected the model with the best F1-score overall, which is 0.65 (61% recall and 69% precision). The model uses 40 compact sentences in context, 0.25 EMR and was trained on both synthetic and human-annotated data. However, note that the hyperparameters of the best models are quite varied. Although compact texts are generally helpful, there are also some well-performing models using simple templates only. A higher number of sentences in context may help to achieve better F1-score, but not always (the longer context is also sometimes cropped to fit the input). Using a higher EMR then generally leads to higher recall, suggesting that the model adapts to the base rate of errors.

4 Results of our Charles-UPF submission

Table 3 shows the results of our model on the official test data of the task, broken down by error types. The overall scores are higher than on the development set – test set recall is 0.691 (vs. 0.614 on the development set) and precision is 0.756 (vs. 0.690). The fact that we used the whole available human annotated data for training the final model may have contributed to the difference, but it is also possible that the test data was somewhat less challenging. We note that our model was able to identify only three types of errors (*NAME*, *NUMBER*, *WORD*), having better results for the *NAME* and *NUMBER* errors. We believe the explanation is two-fold: the names and numbers are often found verbatim in the input data (and in our generated facts), which makes them easy to detect, and also the corresponding error types were the most represented in the training data. In contrast, the three error types which were not detected are much less represented in the training data and hard to detect in our setup.

model trained only on annotated data. The results were overall in the 0.3-0.5 range for both recall and precision, and no model was the best-performing one in terms of any metric.

5 Discussion

Our Charles-UPF submission achieved the best results in the automatic metrics category, but there is still a gap with what humans can achieve, as shown by the Laval University submission's (Garneau and Lamontagne, 2021) overall 0.841 recall and 0.879 precision. One way to improve our system would be to enrich the reference fact descriptions, by either inferring more information from the raw data, or by extracting additional data from external databases.[2] Another option would be to add surrounding sentences to the context – this could help to resolve coreferences (e.g., if a player is referred to as *"He"*) and to detect the *CONTEXT* errors.

We also note that our approach requires the real system outputs manually annotated with errors in order to work well – using only synthetic data results in low recall (see Table 2). However, we believe that more sophisticated techniques for creating the synthetic data could help to achieve same results with less human-annotated data. We also believe that our system is in general applicable to new games or seasons. The rule-based generator does not need any adapting, the vocabulary of both neural parts (context selector and error tagger) is based on subwords and thus also able to handle unseen player/team/city names. The model effectively learns to compare entities from the context and the evaluated sentence, the absolute values are thus less important than their agreements and differences.

6 Conclusion

We presented our system for detecting errors in generated descriptions of basketball matches. Our system can automatically classify the errors on token level, using a pretrained language model and textual description of data generated by a rule-based NLG system. Our system reached 0.691 recall and 0.756 precision on the test data, finishing first out of four automatic metric submissions in the INLG 2021 Accuracy Evaluation shared task.

Acknowledgements

This work was supported by the Charles University projects GAUK 140320, SVV 260575, and PRIMUS/19/SCI/10, an Apple NLU Research Grant for Heriot-Watt University and Charles University, and by the European Commission via UPF under the H2020 program contract numbers 786731, 825079, 870930 and 952133.

References

Anya Belz, Simon Mille, and David M. Howcroft. 2020. Disentangling the properties of human evaluation methods: A classification system to support comparability, meta-evaluation and reproducibility testing. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 183–194, Dublin, Ireland. Association for Computational Linguistics.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional WebNLG+ shared task: Overview and evaluation results (WebNLG+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 55–76, Dublin, Ireland (Virtual).

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy.

Ondřej Dušek, David M. Howcroft, and Verena Rieser. 2019. Semantic noise matters for neural natural language generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 421–426, Tokyo, Japan. Association for Computational Linguistics.

Ondřej Dušek and Zdeněk Kasner. 2020. Evaluating semantic accuracy of data-to-text generation with natural language inference. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 131–137, Dublin, Ireland. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. *Computer Speech & Language*, 59:123–156. ArXiv: 1901.07931.

Katja Filippova. 2020. Controlled hallucinations: Learning to generate faithfully from noisy data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 864–870.

Nicolas Garneau and Luc Lamontagne. 2021. Shared task in evaluating accuracy: Leveraging pre-annotations in the validation process. In *Proceedings of the 14th International Conference on Natural Language Generation*, Aberdeen, Scotland, UK.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2410–2424.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102.

Youngwoo Kim, Myungha Jang, and James Allan. 2020. Explaining text matching on neural natural language inference. *ACM Transactions on Information Systems (TOIS)*, 38(4):1–23.

Guy Lapalme. 2020. RDFjsRealB: a symbolic approach for generating text from RDF triples. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 144–153, Dublin, Ireland (Virtual).

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231.

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368, Tokyo, Japan. Association for Computational Linguistics.

Peiguang Li, Hongfeng Yu, Wenkai Zhang, Guangluan Xu, and Xian Sun. 2020. SA-NLI: A supervised attention based framework for natural language inference. *Neurocomputing*, 407:72–82.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Fixing weight decay regularization in Adam. In *International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA.

Simon Mille, Stamatia Dasiopoulou, Beatriz Fisas, and Leo Wanner. 2019. Teaching FORGe to verbalize DBpedia properties in Spanish. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 473–483, Tokyo, Japan. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA.

Clément Rebuffel, Marco Roberti, Laure Soulier, Geoffrey Scoutheeten, Rossella Cancelliere, and Patrick Gallinari. 2021. Controlling hallucinations at word level in data-to-text generation. *arXiv preprint arXiv:2102.02810*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Ehud Reiter and Craig Thomson. 2020. Shared task on evaluating accuracy. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 227–231, Dublin, Ireland (Virtual).

Sashank Santhanam and Samira Shaikh. 2019. Towards best experiment design for evaluating dialogue system output. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 88–94, Tokyo, Japan. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online.

Craig Thomson and Ehud Reiter. 2020a. A gold standard methodology for evaluating accuracy in data-to-text systems. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 158–168, Dublin, Ireland. Association for Computational Linguistics.

Craig Thomson and Ehud Reiter. 2020b. A gold standard methodology for evaluating accuracy in data-to-text systems. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 158–168, Dublin, Ireland (Virtual).

Craig Thomson and Ehud Reiter. 2021. Generation challenges: Results of the Accuracy Evaluation Shared Task. In *Proceedings of the 14th International Conference on Natural Language Generation*, Aberdeen, Scotland, UK.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2019. Generating token-level explanations for natural language inference. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 963–969, Minneapolis, MN, USA.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*.

Trung Tran and Dang Tuan Nguyen. 2020. WebNLG 2020 challenge: Semantic template mining for generating references from RDF. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 177–185, Dublin, Ireland (Virtual).

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Shehel Yoosuf and Yin Yang. 2019. Fine-grained propaganda detection with fine-tuned BERT. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 87–91, Hong Kong.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations (ICLR)*, Online.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491, Online.

Shared Task in Evaluating Accuracy: Leveraging Pre-Annotations in the Validation Process

Nicolas Garneau and Luc Lamontagne
Université Laval, Québec, Canada
{nicolas.garneau,luc.lamontagne}@ift.ulaval.ca

Abstract

We hereby present our submission to the Shared Task in Evaluating Accuracy (Reiter and Thomson, 2020) at the INLG 2021 Conference. Our evaluation protocol relies on three main components; rules and text classifiers that pre-annotate the dataset, a human annotator that validates the pre-annotations, and a web interface that facilitates this validation. Our submission consists in fact of two submissions; we first analyze solely the performance of the rules and classifiers (pre-annotations), and then the human evaluation aided by the former pre-annotations using the web interface (hybrid). The code for the web interface and the classifiers is publicly available[1].

1 Introduction

Evaluating Data-to-Text natural language generation (NLG) systems is a very important task despite its notorious difficulty (Thomson and Reiter, 2020). Reiter and Thomson introduced a Shared Task in Evaluating Accuracy which consists of factually assessing the accuracy of basketball games summaries produced by an automatic (neural) language generator. The underlying Data-to-Text dataset was originally created by Wiseman et al..

Evaluating the accuracy of a game's summary relies on identifying the errors within the generated text. Borrowing the same terminology as Reiter and Thomson, the set of possible errors comprises 6 different categories; *Number*, *Named entity*, *Word*, *Context*, *Not Checkable*, and *Other*. We refer the reader to the original paper for more details about these different error categories.

As part of the shared task, participants were asked to propose an automatic evaluation metric (or algorithm) and/or an evaluation methodology that could be followed by humans in order to assess the accuracy of a given generated text. In our case, we proposed both, i.e. a hybrid approach where the evaluation methodology leverage pre-annotations in order to accelerate (and hopefully improve) the evaluation process.

[1] https://github.com/ngarneau/accuracySharedTask

2 Evaluating Accuracy

Our evaluation methodology relies on the pre-annotation of game summaries and a validation procedure that we describe in the following sections.

2.1 Pre-Annotation of Game Summaries

In order to accelerate the task of evaluating the accuracy, we propose to pre-annotate the game summaries (i.e. identify *potential* errors) using a set of rules and text classifiers. To this end, we followed the hierarchical proposition specified by Reiter and Thomson[2] and designed our system accordingly.

We then separate the set of errors into two groups. The first group contains *Number* and *Name* errors, where every instance can easily be identified (not necessarily validated) by an algorithm. This includes names beginning with a capital letter and numbers consisting of either digit (1, 2, etc.), written words (one, two, etc.) or ordinals (first, second, etc.). The second group contains all other error types, i.e. *Word*, *Context*, and *Not Checkable* [3]. We illustrate the distribution of errors on the development set in Table 1

For the first group of errors, we designed two simple rules. Given the set of n *Number* instances $\{u_i | i \in 1 \dots n\}$ and m *Name* instances $\{a_j | j \in 1 \dots m\}$ in a given sentence, we check for the following;

[2] *Number* > *Name* > *Word* > *Context* > *Not Checkable* > *Other*

[3] Since the *Other* type of error does not provide many examples and it was designed for nonsensical text, we do not consider it in our submission.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 266–270,
Aberdeen, Scotland, UK, 20-24 September 2021.

Error Type	Count
Number	474
Name	317
Word	334
Context	51
Not Checkable	37

Table 1: Distribution of error types in the development set for the first group (*Number* and *Name*) and the second group (*Word*, *Context*, and *Not Checkable*).

1. For every pair (u_i, a_j), find a correspondence (row–column wise) in the Box Score. If the check fails, we consider it is a *Number* error.

2. For every a_j, find a correspondence (anywhere possible) in the Box Score. If the check fails, we assume it is a *Name* error.

These checks are done to pre-annotate sentences with *Number* and *Name* errors which are later validated by the annotators. We provide more details on the validation step in Section 2.2.

The errors of the second group are much more difficult to identify. Since the annotators might not be native English speakers or might have little knowledge about the basketball lexical field, we help them with two textual classifiers that are trained on the development set released along with the task. The first classifier predicts if a sentence may contain errors belonging to the second group or not (i.e. a binary classification). The second classifier predicts which type(s) of errors may be present within a sentence positively labeled to the second group. Hence it is a multiclass, multilabel classifier. It takes as input only the sentence, no contextual information from the box score. This classifier uses unigrams, TF-IDF weights (Sparck Jones, 1988) and a multinomial logistic regression model. The regression model assigns to each word w_i in the vocabulary a score specific to a class c_j, namely $s_{i,j}$. Using the TF-IDF weight t_i and a classification threshold τ, we classify a word as being erroneous w.r.t. a specific class c_j as follows:

$$c_{i,j} = \begin{cases} 1, & \text{if } t_i \cdot s_{i,j} \geq \tau \\ 0, & \text{otherwise} \end{cases}$$

$c_{i,j}$ is thus used as a pre-annotation to the word w_i. If multiple error labels can be associated with a word, we take the one with the highest importance. We used $\tau > 0.5$ in our experiments.

The binary and multilabel classifiers have been trained using the scikit-learn library (Pedregosa et al., 2011) and achieve respectively 0.63 and 0.76 F1-scores on the development set.

2.2 Evaluation Procedure

Our evaluation procedure is composed of three steps, where the first and the last are fully automated while the second is performed by a human.

Importing games data and pre-annotation. The annotator first imports the games' data into the validation database with a python script. Then, the game summaries are automatically pre-annotated using the classification models described in Section 2.1. The games are then ready to be validated manually.

Validating pre-annotations. The list of games to validate is displayed through the web interface to the annotator. The annotator selects the game to annotate and begins with the first sentence. To provide a little more context, we include links to the box score, to each team's respective pages, and to the current calendar showing the dates from the month in which the game was played. We also present the previous and next sentences to the current one, if there are any. Then, a list of all the words of the sentence under study is presented, with a dropdown list filled with pre-annotations that lets the user select the possible error a given word might be associated with. The annotator is asked to follow the evaluation guidelines provided by Reiter and Thomson. Once the validation is done, the user can save them and move onto the next sentence until the summary is fully validated. We illustrate in Figure 1 the main interface proposed to the annotator.

Preparing submission file. Once all the games have been validated, the annotator needs to prepare a submission file. This file is created using a python script and made available to the evaluators.

3 Results

3.1 Pre-Annotations

We can see from Table 2 that the pre-annotations offer a basic coverage for the *Number* and *Name* errors. Recall that these pre-annotations rely strictly on two simple rules.

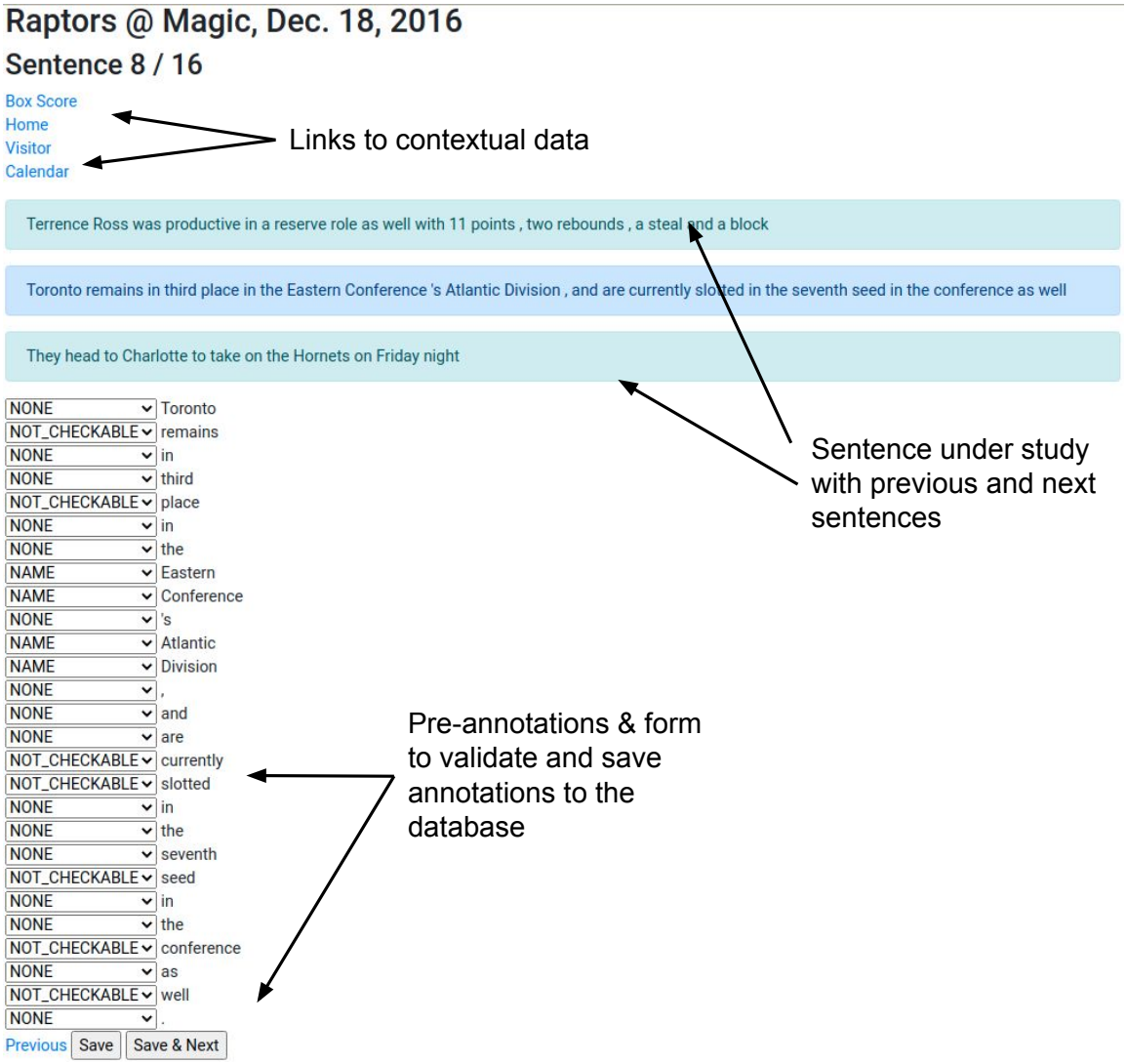


Figure 1: The main interface for the annotator used to validate the pre-annotations. Links to the box score, teams' statistics, and current calendar are available at the top. The previous and next sentences are shown to the annotator to provide more context. The annotator validates the pre-annotations by updating the dropdown lists in the form.

		Pre-Annotations						Human Validation					
	Error	**Num.**	**Name**	**Word**	**Cont.**	**N/C**	**Avg.**	**Num.**	**Name**	**Word**	**Cont.**	**N/C**	**Avg.**
DEV	**Precision**	0.38	0.67	0.20	0.00	0.10	0.31	0.65	0.78	0.33	0.08	0.16	0.49
	Recall	0.30	0.53	0.58	0.00	0.60	0.49	0.58	0.69	0.67	0.16	0.60	0.66
TEST	**Precision**	0.35	0.79	0.14	0.00	0.19	0.33	0.88	0.94	0.73	0.40	0.39	0.88
	Recall	0.44	0.59	0.36	0.00	0.50	0.50	0.86	0.92	0.68	0.75	0.24	0.84

Table 2: Precision and Recall results for every error types on the development and test set. We present the difference between the pre-annotations only, and the human validation.

The classifiers, however, struggle to precisely identify erroneous words (while having a decent recall). During validation of the summaries, we noticed that the classifiers acted more like pointers. Indeed, when there were potential *Word*/*Context*/*Not Checkable* errors, the classifiers flag non-erroneous words. Take for example the sentence in Figure 1. The classifier identified the first words like "remains" and "place" as being *Not Checkable* errors. While these words do not correspond to the exact error ("*third* place"), it does give a pointer to the annotator that there is something to verify within

this particular phrase. Nonetheless, it did help the annotator to give more serious attention to these sentences, and especially to detect *Context* errors.

We can see from Table 2 that there are negligible differences across error types between the development and test set. This suggests that the pre-annotations did not overfit.

3.2 Human Evaluation

The human evaluation was conducted in two subsequent stages; the annotator first validated the development set and then the test set, and a difference in the results obtained for these two data sets can be observed in Table 2. This difference is attributable to the fact that the annotator gained domain knowledge throughout the annotation process. This is especially true for the *Word* and *Context* errors. Overall, on the test set, we achieve interesting (and surprisingly high) precision and recall scores of 0.88 and 0.84 respectively. It would have been interesting to validate the test set with an annotator that did not have any prior knowledge on the task and domain i.e. had not seen the development set.

3.3 Evaluation Time

Familiarisation with the problem, the dataset, and the domain took roughly 4 hours. The development of the evaluation interface, rule modeling, and creation of the classifiers represent an effort of one day (8 hours). The training time of the classifiers is near-instantaneous, and the computing power needed is CPU only. The evaluation time required for one game is between 10 to 15 minutes. Since we save the normalized annotations within the database, it is easy to generate the submission file with a simple script that takes seconds to run. Overall, considering the validation of both development and test sets, conducting the whole experiment took around 30 to 35 hours. Originally, the protocol took around 30 minutes per annotator and used 3 relatively knowledgeable annotators to achieved good results. We thus considerably reduced the manual effort.

4 Discussion

As previously mentioned, evaluating the accuracy of generated text in a Data-to-Text setting is a very hard task. In this experiment, the annotators found the task especially difficult since they are not native English speakers and basketball game descriptions are not (or were not!) a domain with which they were familiar.

While the following example may seem trivial, it took several summaries for the annotator to understand that "led the bench" means that it is targeting non-starting players (one could easily assume that every player starts on the bench). Also, an annotator who knows which player plays for which team will greatly improve and accelerate the evaluation process. This is something that the annotator just barely began to get comfortable with towards the end of the task.

Our experiments, due to the lack of time and resources, do not explicitly expose the benefits of having pre-annotations. It would have been interesting to see if, given two annotators with the same prior knowledge[4], the pre-annotations helps them to solve the task both in a matter of time and accuracy, hence evaluating the inter-annotator agreement. Nonetheless, the pre-annotations did help the annotators through the evaluation process in three different ways;

1. get familiar with the task, domain, and dataset by seeing which potential word may be erroneous;

2. identify errors that the annotator would have missed;

3. save time and effort.

We decided to design our own annotation tool instead of using WebAnno (Yimam et al., 2013), the one initially used by the authors of the task. This decision was mainly motivated by the flexibility of storing and pre-annotating the game summaries. In future works, we would consider more robust text classifiers coupled with active learning in order to improve the pre-annotations, while executing the task. We would also consider adding contextual information from the box score for the classification.

References

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[4]Then again, it is hard to measure *prior knowledge* on a specific domain.

Ehud Reiter and Craig Thomson. 2020. Shared task on evaluating accuracy. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 227–231, Dublin, Ireland. Association for Computational Linguistics.

Karen Sparck Jones. 1988. *A Statistical Interpretation of Term Specificity and Its Application in Retrieval*, page 132–142. Taylor Graham Publishing, GBR.

Craig Thomson and Ehud Reiter. 2020. A gold standard methodology for evaluating accuracy in data-to-text systems. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 158–168, Dublin, Ireland. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. WebAnno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria. Association for Computational Linguistics.

Automatic Verification of Data Summaries

Rayhane Rezgui
EURECOM, France
Rayhane.Rezgui@eurecom.fr

Mohammed Saeed
EURECOM, France
Mohammed.Saeed@eurecom.fr

Paolo Papotti
EURECOM, France
Paolo.Papotti@eurecom.fr

Abstract

We present a generic method to compute the factual accuracy of a generated data summary with minimal user effort. We look at the problem as a fact-checking task to verify the numerical claims in the text. The verification algorithm assumes that the data used to generate the text is available. In this paper, we describe how the proposed solution has been used to identify incorrect claims about basketball textual summaries in the context of the Accuracy Shared Task at INLG 2021.

1 Introduction

Natural Language Generation (NLG) can be used to generate texts out of relational data, where the goal is to have correct and clear statements that describe what can be found in tables (Gong et al., 2019). However, this is not always the case, since NLG tools, although producing correct sentences grammar-wise, sometimes fail to generate accurate texts, eventually containing some factual errors (Wiseman et al., 2017).

The goal of our work is to evaluate generated texts by identifying numerical claims and fact-checking them with the relational data available at hand. We apply different techniques on the provided summaries and use the available relational data about the matches to state whether the claims are true or false. The idea behind fact checking using relational tables is to create an automated verification pipeline using data-driven algorithms, such as deep learning models (Nakov et al., 2021; Saeed and Papotti, 2021). Building upon previous work in fact-checking statistical claims (Karagiannis et al., 2020), we focus on such kind of claims due to the availability of trustworthy relational tables to verify them. The goal is not only to be accurate, but also to limit the user effort, such as labelling data or writing scripts, in the setup of the system.

The Memphis Grizzlies (5-2) defeated the Phoenix Suns (3 - 2) Monday 102-91 at the Talking Stick Resort Arena in Phoenix. The Grizzlies had a strong first half where they out-scored the Suns 59-42. Marc Gasol scored 18 points, leading the Grizzlies. Isaiah Thomas added 15 points, he is averaging 19 points on the season so far.

Figure 1: Example of a generated textual summary from the basketball relational data.

Team	PTS	AST	..	TOV
Kings	99	22	..	21
Nets	107	20	..	9

(a) Sample of team statistics in a match.

Team	PTS	AST	..	FGA
Ben McLemore	11	2	..	9
Sergey Karasev	5	3	..	5
..	..	..	..	..
Kevin Garnett	10	2	..	10

(b) Sample of player statistics in a match.

Statistical claims form a significant part of the set of claims in the shared task (around 40%) as the tables mostly contain numerical facts, rather than textual. Consider samples of the relational tables of players and teams in Tables 2a and 2b, respectively. Sentence "Sacramento Kings scored 99 points." is verified by identifying the team name (key value "Kings") and column (label "PTS"). A comparison between the value in the identified cell and the value in the text (99) validates the claim. More complex claims, such as "Kings defeated Nets", require comparisons between two cells.

To verify the claims above, our solution consists of three main steps: (i) claim identification, (ii) identification of properties that construct a val-

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 271–275,
Aberdeen, Scotland, UK, 20-24 September 2021.

idation query over the data for every claim, and (iii) claim verification. In the following, we first describe our solution and then report some experimental results and possible directions for future work.

2 Method

We target claims that can be verified computationally by using operations over the cell values in the tables (relations). We follow an approach inspired from previous work on computational fact-checking for statistical claims (Karagiannis et al., 2020). We begin the process by extracting claims from the input sentences (Section 2.1) and then identify query properties (Section 2.2), which are used for building the query that verifies the claim (Section 2.3).

Figure 3 represents the architecture of the solution, where we input sentences and get a collection of properties, including the claims to verify and the elements of the data that are needed for this task. Every sentence can contain more than one named entity (were we focus only on players/teams) as well as several claims. We have to associate each claim and the resulting properties to the entity in question. By looking at the summaries in the task, it is possible to observe that many sentences fall into two categories:

- *Comparative sentences* where the text describes both teams and compares the scores from both sides, e.g., "The Sacramento Kings defeated the Brooklyn Nets 107 - 99." In this case, we assume an order in the sentence. The first claim to be extracted is the one we associate to the first entity. We enumerate all the numerical claims and assign them to the first or second entity based on first appearance. For our example, 107 is assigned to Kings and 99 to Nets. As for the "defeated" claim, we associate it to both Kings and Nets.

- *Look-Up sentences* where the text specifies information/statistics about one entity. In this case, we just define that entity as the row of interest. Otherwise, we associate our claims to the first player to appear.

After claim and properties are identified, we use the latter to create a query that fact checks the claim. The query returns a Boolean value in the final output and it is self-explanatory, i.e., it is a declarative specification easy to interpret as an explanation of the checking process. An example of a query that returns the number of **points (PTS)** of the team **Sacramento Kings** from an identified table **t** is

```
SELECT t.PTS FROM t WHERE
t.Team='Kings';
```

whose output is compared against the value of the extracted claim.

2.1 Claim Identification

Following the data-generation procedure in (Karagiannis et al., 2020), we wrote 9 templates to generate natural-language sentence starting from the provided tables. These scripts could be replaced with NLG algorithms in a fully automatic solution (Parikh et al., 2020). The generated data is used to fine-tune a **BertForTokenClassification** model (Devlin et al., 2019) to identify claims in an input sentence. More precisely, the input sentence is tokenized and a binary-classification layer is applied on every token to predict whether the token is part of the claim or not. As some claims occur together, we rely on textual separators (like commas) to separate them. The following sentence shows an example of a sentence with six claims underlined: "AJ Hammons had **5 rebounds**, **10 assists**, **6 turnovers**, **7 points**, **3 steals** and **1 block**."

The model is able to correctly identify claims on the synthetic test dataset. However, we also tried another simpler regex-based approach where we focus on key words like the columns (points, turnovers, etc.) and extract the neighboring words. Both approaches gave similar results on sentences sampled from the dataset of the shared task, with BERT failing to extract the correct claims for some sentences. This is mainly due to two reasons. First, neural models are frail w.r.t. small changes in textual input (Zhang et al., 2020). Second, when multiple claims occur sequentially, there is no clear way of separating them unless some form of separator is found (like a comma). A simple fix to this would be to include spaces in the tokenized input and learn which spaces are included in a claim and which are not. The regex solution is pragmatic in cases where annotated data is not available as it produces results comparable to the classifier solution.

The regular expressions for claim identification check for trigger words (such as 'assists', 'blocks', 'field goals', 'minutes', 'points','rebounds', 'steals', 'turnovers') and then identify the following string in the text. This string can be a numerical value in its

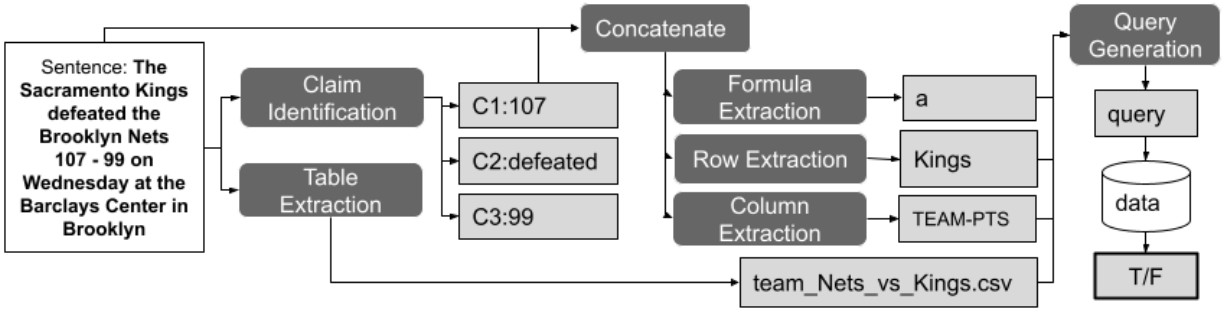


Figure 3: Architecture of the solution. Given a sentence, we identify the claims and the table to verify them. Then every claim (C1-C3 in the example) is processed individually to obtain a query for its verification. (We take claim C1 as an example in the figure.)

numerical format (15) or in words (fifteen). These expressions might identify some false positives. In the sentence "The only other Net to reach double figures in points was Ben McLemore", we identify a string ("in points") which cannot be verified as it does not contain a numerical value. We filter out such claims before verification. For comparison sentences, we consider "defeated", "outscored" and "lost" in the word list.

2.2 Property Identification

After identifying claims in the text summary, we ought to predict the data and the operation that allow us to verify them. We call *properties* the elements that ultimately enable the generation of a verification query for the given claim. Properties include: (i) the name of the table (relation), (ii) the primary key value (i.e., the identifier for a tuple in the relation), (iii) the column (i.e., the attribute label), as well as (iv) the formula, which include the simple look up of a value (a) and value comparisons (b>a, a>b). We describe the main modules, as depicted in Figure 3 next.

For the *column* and the *formula* extraction, we fine-tune a *BertForSequenceClassification* model (Devlin et al., 2019) with generated training data in Section 2.1. For the column identification, we have 15 possible classes, whereas for the formula identification we only have 3 classes with most examples pivoting on the "lookup class" (a). This is due to the fact that most relevant statistics are already reported in the tables, such as ratios (a/b), and they can be verified with a simple lookup.

For a given input text, extracting the team names is crucial as they enable identifying the *table* and primary *key value*(s) of a sentence. The given textual inputs describe a single basketball match which usually begin with a general sentence mentioning the associated teams. We therefore search in the first sentence of a given text for the mentioned team names. For every pair of teams, (i.e., for every match), we consider one table reporting the team statistics (Figure 2a) and another one reporting player statistics (Figure 2b). These are used later on for other sentences in the same summary, where team/player names are identified accordingly to the associated tables.

There are sentences where both team and player names occur. In this case, we use the player name as the primary key value, since claims are more likely to be related to the player than to the team. Out of 257 sentences that were extracted from the test files, 22 sentences contain more than 2 names, with at least one of them being a player name, such as "**Bradley Beal** led the way for the **Wizards** with a game - high 18 points , which he supplemented with five rebounds , four assists and one steal". Most of these 22 sentences follow the same structure (player led team, team was led by a player), while only one sentence raises an issue, because it has 2 player names rather than one[1].

2.3 Claim Verification

After getting all the elements we need for the verification, we build queries to look up the data tables. Since we organize the data in csv files of the same format (player_TeamA_vs_teamB.csv or team_TeamA_vs_teamB.csv), the queries share a fixed structure. Once the table name has been identified, query generation is driven by the formula obtained from the classifier. We then collect the value(s) for the check by identifying the cell values based on the key and the column predictions; such

[1] "Sergio Rodriguez was the high-point man for the 76ers, with 18 points and five assists, while Jahlil Okafor logged 20 minutes off the bench."

values are finally compared against the claim values. The claim value is translated to a numerical value if written in words (thirteen to 13), or considered True by default in case of a Boolean claim ("defeated").

Claim	Column	Formula	Row
18 points	PTS	a	Bradley Beal
five rebounds	REB	a	Bradley Beal
four assists	AST	a	Bradley Beal

Table 1: Extracted properties for an example sentence.

The following example walks through a sentence verified by our solution. Given "Bradley Beal led the way for the Wizards with a game-high 18 points, which he supplemented with five rebounds , seven assists and one steal.", we show the extracted claims and properties in Table 1.

Claim	Query Output	Evaluation
18 points	18.0	True
five rebounds	5.0	True
seven assists	4.0	False

Table 2: Evaluation of the example.

All claims require a lookup (formula is **a**) in the table player_Hornets_vs_Wizards.csv, with a primary key value **Bradley Beal** and columns **PTS**, **REB**, and **AST**, respectively. After extracting these properties, we compose and execute the final query over the table and compare with the actual claim as shown in Table 2. Our system did not identify claims such as "led" & "game-high" as our claim-identification module is limited by a regular-expression approach.

3 Results

The evaluation of our solution over the test data shows that we obtain a precision of 0.329 and a recall of 0.205. While we expected a limited recall, as we focus on a specific subset of claims among all the possible ones in the summaries, we investigated the possible causes for the low performance and found three main explanations.

Missing support for coreference resolution. In some sentences, names are not explicitly mentioned and the concerned entity is referred to as "he" or "the visiting team". As we did not implemented a specific solution for coreference resolution, this is a weak point in our system. For example, consider the sentence "It was his second double-double in a row, as he's combined for 54 points and 13 rebounds over his last two games.". The sentence is not checked by the system since no names were singled out, leading to the system missing 3 claims.

Claims requiring complex retrieval. Some statistical values are hard to fact check, such as the number of wins in the season. Consider the sentence "Over his last three games, he's combined for 34 points, 13 rebounds and five assists, while playing just 21 minutes per game.'. The verification of this claims requires to identify the last 3 games played by the player, and to sum up their scores. As another example, "Greg Monroe was the only other Laker [...]". The only way to know if he was the only other Laker to achieve something is to look at all the Lakers players scoring. These kinds of processes go beyond the ability of our data retrieval modules.

Limits in the identification of the claims. Some of the wording used in the text turned out to be hard to process both with the regex function and the Bert model. For example, all claims containing an expression like "double-double" or "6-for-13".

4 Future Work

We presented a solution for verifying statistical data claims in data summaries with limited human supervision. We believe our work shows some of the opportunities and challenges for the problem of verifying data summaries with computational methods (Saeed and Papotti, 2021). While there are some promising results for specific cases, the road to accurate and general solutions is still long. The main challenge lies in the limited amount of training data and the large variety of claim kinds, as we discuss next.

The claims in Figure 1 vary from those expressed using adjectives (e.g., strong) to others pivoting on verbs (e.g., leading). Fact-checking these claims requires to go beyond a lookup in a table, but rather, we seem to suggest the need for domain-specific rules or models, depending on the nature of the claim. For instance, for the sentence "the Grizzlies had a **strong** first half", if we interpret that the claim "strong" relates to the points scored in the first half, it means that we are interested in the sum of the first two quarters, *TEAM-PTS_QTR1* + *TEAM-PTS_QTR2*, and we have to set a threshold to the number of points starting from which the word "strong" applies. But this rule would apply

only to "strong" (and similar adjectives) used for points and the threshold would change for "fouls". In other words, we have to handle all the possible qualitative adjectives that can appear in the text for all attributes, which is challenging to do exhaustively and accurately. It is easy to see that this problem is challenging as it would either require a lot of manually defined rules or the annotations of a large number of claims to train models that handle all cases.

References

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

Heng Gong, Xiaocheng Feng, Bing Qin, and Ting Liu. 2019. Table-to-text generation with effective hierarchical encoder on three dimensions (row, column and time). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3143–3152, Hong Kong, China. Association for Computational Linguistics.

Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. 2020. Scrutinizer: A mixed-initiative approach to large-scale, data-driven claim verification. *Proc. VLDB Endow.*, 13(11):2508–2521.

Preslav Nakov, David P. A. Corney, Maram Hasanain, Firoj Alam, Tamer Elsayed, Alberto Barrón-Cedeño, Paolo Papotti, Shaden Shaar, and Giovanni Da San Martino. 2021. Automated fact-checking for assisting human fact-checkers. In *IJCAI*, pages 4826–4832. ijcai.org.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of EMNLP*.

Mohammed Saeed and Paolo Papotti. 2021. Fact-checking statistical claims with tables. *IEEE Data Eng. Bull.*, 44(3).

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(3).

Grounding NBA Matchup Summaries

Tadashi Nomoto
National Institute of Japanese Literature
Tachikawa Tokyo 190-0014, Japan
nomoto@acm.org

Abstract

The present paper summarizes an attempt we made to meet a shared task challenge on grounding machine generated summaries of NBA matchups.[1] In the first half, we discuss methods and in the second, we report results, together with a discussion on what feature may have had an effect on the performance.

1 Introduction

What led Reiter and Thomson (2020) to launch a shared task competition in 2020 was a concern that fact-checking automatically generated texts (machine texts, or M_TEXTs) in the context of data to text generation (Wiseman et al., 2017), is hugely labor intensive, making it virtually impossible to run it at a scale. In an effort towards putting it under control, the project asks participants to find a way to do the assessment automatically, without any human intervention. The problem is set out as follows: you receive M_TEXTs, along with other external information such as box scores and human created summaries (or H_TEXTs). Your goal is to locate factual mistakes in M_TEXTs and classify them according to a pre-defined scheme of error types ('*word,*' '*number,*' '*name,*', '*context,*' '*not checkable*').

2 Method

The following sections detail our approach, which in essence is a multi-pronged strategy. We deploy separate mechanisms to deal with different types of error.

2.1 Detecting Word/Name Errors

We split an M_TEXT into three parts, LEAD, MIDDLE, TAIL (Figure 1), and use a separate set of rules targeting a particular part of the text, to identify errors with word or name.

[1] https://github.com/ehudreiter/accuracySharedTask.git

2.1.1 Lead Section

For the lead section, we focus on date (day of week, DOW) and venue, in particular those located in the first 3 sentences of an M_TEXT. We compare each sentence (call it an m-sentence)[2] in the lead against names of US basketball arenas listed in Wikipedia[3] to get one most similar (based on how much they overlap) and use it as a canonical name. We locate a date expression by going through each token in an m-sentence and pick one that best matches a DOW name we prepared beforehand. We report a name error if there is any conflict between M_TEXT and H_TEXT in DOW or in venue. We do not work with a full sentence. Rather, we work with a *clause*, a minimal sentential unit that serves a building block of a complex sentence.[4] This is meant to ensure that we have no more than one occurrence of a venue and a date in an input we feed to the process. We call a clause contained in 'm-sentence,' an m-clause and that in h-sentence (see Fn. 2), an 'h-clause.' See Algorithm 1 for a more precise picture of what we do here. **search**(X, Y) goes over each of strings given in X to tell if it exists in Y.

2.1.2 Middle Section

In this part, we intend to determine whether a state of affairs described by a cue word holds up, by querying box-office scores. Cue words include words like 'next,' 'led,' 'bench,' and 'defeated,' which make a specific verifiable statement about players and teams. We go through each sentence, to see if it has a player name together with a cue

[2] Similarly we mean by 'h-sentence' a sentence that occurs in H_TEXT.

[3] https://en.wikipedia.org/wiki/List_of_National_Basketball_Association_arenas

[4] We identify and isolate clauses by breaking up a sentence using a dependency tag 'mark' provided by spaCy (https://spacy.io/). For details on what the tag means, consult https://universaldependencies.org/docs/en/dep/mark.html.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 276–281,
Aberdeen, Scotland, UK, 20-24 September 2021.

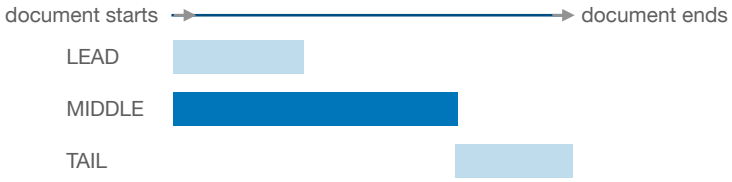


Figure 1: We apply rule based heuristics to different parts of the text to identify errors.

Table 1: Cue Words

single-word cues	*next, bench, reserve, starter, led, leader, leads, best, paces, pace, pacing, paced*
multi-word cues	*player of the game, team - high, high - point, double figures, double - double, triple - double*

Algorithm 1 Finding a name error in LEAD

Input: m-clause, h-clause, DOWs, Venues
Output: **True** or **False**

$H \leftarrow h$-clause ▷ String
$M \leftarrow m$-clause ▷ String
$\mathcal{D} \leftarrow$ DOWs ▷ Pre-def. List of Strings
$\mathcal{V} \leftarrow$ Venues ▷ Pre-def. List of Strings
$d_h \leftarrow$ **search**$(\mathcal{D}, H)$ ▷ returns a date in H
$d_m \leftarrow$ **search**$(\mathcal{D}, M)$ ▷ returns a date in M
$v_h \leftarrow$ **search**$(\mathcal{V}, H)$ ▷ returns a venue in H
$v_m \leftarrow$ **search**$(\mathcal{V}, M)$ ▷ returns a venue in M
if $d_h \neq d_m$ **or** $d_h \neq d_m$ **then**
  return False
else
  return True
end if

Algorithm 2 Finding a word error in MIDDLE

Input: m-clause, Box-Scores, Cue Words, Player Names
Output: **True** or **False**

$S \leftarrow m$-clause ▷ String
$\mathcal{X} \leftarrow$ **pd_load**(Box Scores) ▷ Load into Pandas
$\mathcal{C} \leftarrow$ Cue Words ▷ Pre-def. List of Strings
$\mathcal{P} \leftarrow$ Player Names ▷ Pre-def. List of Strings
for each $c \in \mathcal{C}$ **do**
  for each $p \in \mathcal{P}$ **do**
    if match(s, S) & **match**(p, S) **then**
      $T \leftarrow$ **ask_pandas**$(c, p, \mathcal{X})$ ▷ Ask Pandas if $\mathcal{X}$ supports what c says about p.
      return T
    end if
  end for
end for

word. If found, we go to the box score to decide whether it supports the statement. For example, if the text says that player A is off the bench, we know that for it to be true, the player should not be listed under starter. Or if the text states that the team is led by player A, it has to be the case that the player scored the most points. We flag the statement as correct or incorrect depending on whether it is supported by the box-office scores.

Listed in Table 1 are cue words we used, each of which indicates a particular state of affairs that can be checked with the box scores (which we have done using Pandas.)[5] We also break a sentence where possible into clauses (see Fn. 4). Algorithm 2 gives a general idea of how the process works. **match**(X, Y) is a boolean function that holds true if X is found in Y. We load the box scores into a Pandas' data frame prior to the loop operation. **ask_pandas** handles a query for the data frame, returns true if it finds a piece of data that matches the query and false if not. The code shown in Table 2, for instance, asks whether a player started off the bench.

2.1.3 Tail Section

For this part, our goal is to see if there is any error about future matchups. We gather matchup information, such as date (day of week), home name, visitor name from the last two sentences of M_TEXT and check them against a corresponding part of H_TEXT. Specific operations involved are shown in Algorithm 3. **find_matchup** looks for home name, visitor name and date in a clause given as input. It works on both m- and h-clause.

[5] https://pandas.pydata.org/

Table 2: A code in Pandas

data_frame.loc[['START_POSITION'],[player_name]].values.flatten()[0]

Algorithm 3 Finding a name error in TAIL

Input: m-clause, h-clause, DOWs, Team Names
Output: **True** or **False**

$H \leftarrow h$-clause ▷ String
$M \leftarrow m$-clause ▷ String
$\mathcal{D} \leftarrow$ DOWs ▷ Pre-def. List of Strings
$\mathcal{N} \leftarrow$ Team Names ▷ Pre-def. List of Strings
$m_a, m_b, m_c =$ **find_matchup**$(M, \mathcal{N}, \mathcal{D})$
▷ m_a, m_b, m_c represent home name, visitor name, date found in m-clause, respectively
$h_a, h_b, h_c =$ **find_matchup**$(H, \mathcal{N}, \mathcal{D})$
▷ h_a, h_b, h_c represent home name, visitor name, date found in h-clause, respectively.
if $m_a \neq h_a$ **and** $m_b \neq h_b$ **and** $m_c \neq h_c$ **then**
return **Not_Checkable**
end if
if $m_a \neq h_a$ **or** $m_b \neq h_b$ **or** $m_c \neq h_c$ **then**
return **False**
else
return **True**
end if

In case the search is successful with m-clause but not with h-clause (meaning that none of the targets was found in h-caluse), we stop, reporting that they are unverifiable or uncheckable. Otherwise, we look for a discrepancy between triplets in m- and h-clause, and report an error if any is found.

We collectively call a set of rules we brought together for detecting word/name mistakes, 'WED,' hereafter.

2.2 Detecting Number Errors

2.2.1 Building Training Data

In detecting number errors, we essentially rely on data_utils.py[6] (UTL, hereafter) which extracts from the Rotowire dataset, what we call 'relation quadruples' (relQs), each of which contains information on who scored what points in what category.[7] Having relQs at hand is a useful first step towards error detection as they can tell us where to look for potential errors. For example, given a sentence "*Marco and Spencer came off the bench to combine for 31 points, eight rebounds and 10 assists as well.*", UTL would output relQs like those shown in Table 4. OFFSET indicates where the relevant number starts in the sentence.

We recognize however two problems with UTL: (1) it allows a number to get associated with more than one relation; (2) it could fail to assign any relation at all. Our plan is to avoid these annoyances by bootstrapping UTL with a neural model to predict a correct relation given a player name, a number and a context, i.e. a sentence, in which they occur.

In a move in this direction, we transform relQs into source-label pairs of the form shown in Table 3. The process involves acquiring an m-sentence where a relQ comes from, replacing a player name with '@' and a target number (one for which we are trying to find a relation) with '#,' with all other numbers reduced to '⟨NUMBER⟩.'

In addition, we made sure that each relQ we use for training is supported by the box-office scores, that is, evidence exists in the box scores that demonstrates the veracity of the relQ. This means that we accept relQs in Table 4 as training data only if there are records in the box scores showing that Macro had 31 points, 8 rebounds, and 10 assists. If not, they are all discarded. Also dismissed are relQs where a number occurs ahead of a player name (Table 5).

Moreover, in case a number gets assigned to more than one relQ, the preference is given to one that is consistent with a word that immediately follows that number (shots, rebounds, assists). For example, if we have a sentence '*Macro led the team with a spectacular output of 31 points.*' for which UTL may give ('Marco', OFFSET_0, '31', 'PTS') and ('Marco', OFFSET_0, '31', 'AST'), we will take the first relQ and drop the second, as it contradicts what the sentence says about how the number came about (it is not about how many assists he made).

[6] https://github.com/harvardnlp/data2text/

[7] UTL works by locating a player name and a number in a sentence and searching box office scores for records that match the name and the number. It returns all the matches, together with relevant categories, e.g. points, rebounds, assists, steals, blocks, threes, field-goal percentage, free-throw percentage, etc. If the search fails, it returns a relQ with a category named 'NONE.' Throughout the paper, we refer to categories as *relations*, following Wiseman et al. (2017).

Table 3: Source Label Pairs. '@' is a proxy for a person name and '#' that for a numeral of interest.

SOURCE	LABEL
@ and Spencer came off the bench to combine for # points , ⟨NUMBER⟩ rebounds and ⟨NUMBER⟩ assists as well .	PTS
@ and Spencer came off the bench to combine for ⟨NUMBER⟩ points , # rebounds and ⟨NUMBER⟩ assists as well .	REB
@ and Spencer came off the bench to combine for ⟨NUMBER⟩ points , ⟨NUMBER⟩ rebounds and # assists as well .	AST

Table 4: Relation Quadruples, each composed of player name, location, number (points), and label (i.e. category in which points are earned).

('Marco', OFFSET_0, '31', 'PTS')
('Marco', OFFSET_1, '8', 'REB')
('Marco', OFFSET_2, '10', 'AST')

Table 5: Player name has to appear ahead of number. 'w' represents an arbitrary word.

ALLOW	DISALLOW
w w @ w w # w w w	w w w w w # w @ w
@ w w w w # w w w	w w w w w # @ w w
w @ w w w # w w w	w w w w w # w w @

2.2.2 Model

The training data are fed into an LSTM-based Sequence to Label classifier (bidirectional, batch-normalized with the RELU non-linearity):

$$o = \text{softmax}(r(\ell_2(r(\ell_1(m(\mathbf{W})) \qquad (1)$$

$\mathbf{W}$ is an input (a sequence of words that represents a sentence (see Table 3)) where each token is replaced by a word embedding from GloVe,[8] $r(\cdot)$ denotes the RELU activation, $\ell(\cdot)$ a fully connected layer and $m(\cdot)$ a bidirectional LSTM, all of which were built with PyTorch.[9]

After processing the test set in the same way as we did with the training set, we run the model (Eqn. 1), making a prediction about the relation for each relQ instance we find in the text. We label a relQ instance as wrong if it is predicted to have a relation inconsistent with one given by UTL.[10] We refer to the model described here as 'NED.'

3 Resolving Coreference

Given the way UTL works, it is important that we make explicit what a referring expression points to, in order for UTL to successfully build a relQ. To this end, we make use of NeuralCoref 4.0,[11] which operates as an add-on functionality for spaCy.[12] Resolving coreferences with NeuralCoref (NC) results in every referring expression (r-expression, hereafter) in a text being replaced with a corresponding root entity (i.e. its canonical name). This can be troublesome though, because it may disrupt the way in which words are originally laid out, which we need to retain in order to report results conformant to the shared task format policy (which asks to report errors by indicating where they are in the original position). In response, we pursued an approach where we represented a text with a linked list structure in which each word is represented as a node which contains information on what node it is preceded by and what it is followed by, in addition to where it occurs relative to others.[13] For each r-expression NC found, we replaced a token string held by a relevant node with its antecedent while keeping other information (occurrence site, forward/backward connections) in tact (Fig. 2). Furthermore, we restricted an r-expression subject to replacement, to be among 'their,' 'they,' 'he,' 'his,' 'its,' 'it,' and 'him.'

4 Setup and Results

The training data that NED used are sourced from part of the Rotowire corpus (Wiseman et al., 2017),[14] called 'train.json,' which contains 3,398 matchup results each with a summary manually

[8] https://nlp.stanford.edu/projects/glove/
[9] https://pytorch.org/
[10] For instance, we take the following situation as mistake.
UTL ouptut: ('Marco', 0, '31', 'NONE')
Prediction: ('Marco', 0, '31', 'PTS')
[11] https://github.com/huggingface/neuralcoref.git
[12] https://spacy.io/
[13] A node is a structure schematically defined as:
node := ⟨word-token, preceded-by, followed-by, position⟩
[14] https://github.com/harvardnlp/boxscore-data.git

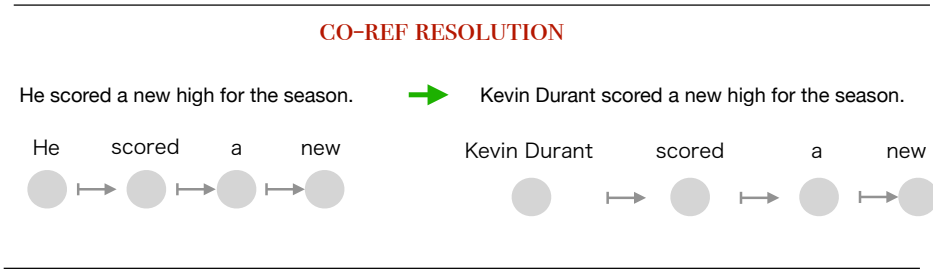


Figure 2: Doing coreference resolution without disrupting the token position

Table 6: Comparison of Per-Type Performance: Coref. vs. Non-Coref. (Shared Task 2021 Train)

	Coref.				Non-Coref.			
	Mistake		Token		Mistake		Token	
	recall	precision	recall	precision	recall	precision	recall	precision
name	0.356	0.958	0.259	0.958	0.356	0.958	0.259	0.958
number	0.781	0.561	0.764	0.561	0.641	0.476	0.628	0.476
word	0.398	0.364	0.311	0.394	0.398	0.364	0.311	0.394
context	0.000	-	0.000	-	0.000	-	0.000	-
not checkable	0.000	-	0.000	-	0.000	-	0.000	-
macro summary	0.549	0.556	0.410	0.553	0.496	0.511	0.375	0.514

Table 7: Results on Shared Task 2021 Test

	Mistake		Token	
model	recall	precision	recall	precision
WED/NED	0.523	0.494	0.349	0.505

created, providing 116,579 source-label pairs in total. The official test set, carved out of the Rotowire corpus, contained machine generated 30 matchup reports and associated box records. We trained NED for 50 epochs on the extracted pairs, achieving the classification accuracy of 0.461 on the test set (over 34 labels). We ran WED on summaries included in the test set (M_TEXTs). It also made use of human summaries provided as part of the Rotowire data (H_TEXTs). Results on Accuracy Shared Task 2021 Test (30 M_TEXTs) are shown in Table 7. The figures are in macro precision and recall. Table 6 gives a per-type comparison of coreference enabled (CrE) versus disabled (CrD) approach, on the train part of 2021 Shared Task (60 M_TEXTs), which reveals a clear advantage of CrE over CrD in the number category (highlighted in blue and red).[15]

[15]It implies that NeuralCoref, by making NED more accurate in predicting relations in relQs, may have pushed higher the system's performance in detecting number errors. We were not able to find any meaningful difference between CrE and CrD on the 2021 Test.

5 Conclusion

This paper gave an overview of the approach we took to meet the shared task challenge for 2021, which is essentially a combination of hand crafted rules (WED) and machine learning (NED): it relied on rule based heuristics to identify errors in name and word while bringing in a neural model to locate number errors. The rule based part consisted of 'translating' a cue expression into a procedure to query box scores for its veracity, while the machine learning part was driven by a neural model, whose predictions allowed us to detect inconsistencies in number related statements in M_TEXTs.[16]

[16]Response to Organizers' Question (which is about how we would cope with unknown player names, venues and teams). Bringing up to date the database we use for WED, of players, teams and venues through Wikipedia and sources devoted to NBA conferences (e.g. box-office scores), could lessen a possible negative impact due to the lack of exposure to data not available at the time of training. If that does not work, we may go to some off-the-shelf NE tool such as one by spaCy, though we expect it may hurt name error detection (due to its reduced accuracy), and to a lesser degree word error detection as it does not involve the recognition of venue names (player and team names can easily be picked by looking at box-office scores which we assume are available all the time). Cue expressions we used were fairly generic. It is highly unlikely that they become less effective on summaries beyond the training data, though we recognize the need to expand the list.

References

Ehud Reiter and Craig Thomson. 2020. Shared task on evaluating accuracy. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 227–231, Dublin, Ireland. Association for Computational Linguistics.

Sam Wiseman, Stuart M. Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation.

Reproducing a comparison of hedged and non-hedged NLG texts

Saad Mahamood
trivago N.V.
Düsseldorf, Germany
saad.mahamood@trivago.com

Abstract

This paper describes an attempt to reproduce an earlier experiment, previously conducted by the author, that compares hedged and non-hedged NLG texts as part of the ReproGen shared challenge. This reproduction effort was only able to partially replicate results from the original study. The analysis from this reproduction effort suggests that whilst it is possible to replicate the procedural aspects of a previous study, replicating the results can prove more significantly challenging as differences in participant type can have a potential impact.

1 Introduction

There has been within recent years a great interest in understanding and quantifying the reproducibility of experiments across several areas of scientific research. This also includes experiments in the field of Natural Language Understanding (NLU), where researchers have questioned the degree to which experiments and results can reliably be reproduced. Recent working exploring the reproducibility of past NLU work has found significant issues such as only a minority of systems reproducing previously reported scores and systems not working due to non-functional code or resource limits (Belz et al., 2021). Additionally, there has been growing awareness of systematic issues with regards to how human evaluations are being conducted. In particular, the lack of standardisation and significant under reporting of key human evaluation details (Howcroft et al., 2020). These twin concerns has led to the creation of the ReproGen shared task (Belz et al., 2020), which attempts to check the reproducibility of human evaluations within the field of Natural Language Generation (NLG).

As part of the ReproGen shared task[1], a reproduction experiment was attempted for a previous work that the author had previously conducted in 2007. In this previous work a human evaluation was conducted between NLG texts containing hedge phrases and those that do not (Mahamood et al., 2007). This past experiment was conducted to better understand the impact of hedge phrases can have when introduced into a data-to-text NLG system. This was done in order to understand how such systems should communicate potentially emotionally sensitive information to a given reader.

In this paper we will describe the experimental setup used and the differences that were made in the reproduction experiment (Section 2), the results obtained and how they compare to the ones originally obtained (Section 3), and finally we will discuss the significance of the results obtained in this reproduction effort (Section 4).

2 Experimental Setup & Differences

2.1 Procedure

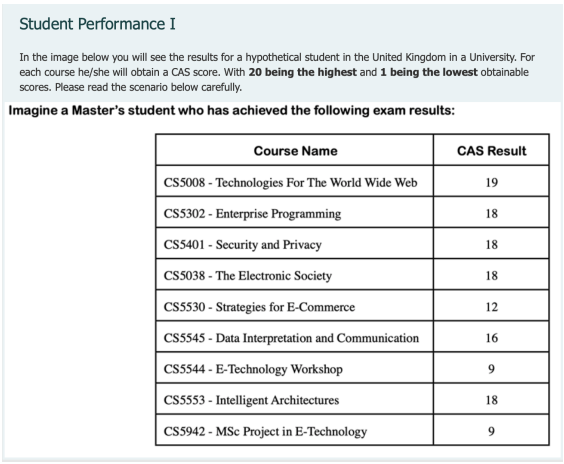
Like the previous experiment, this reproduction experiment sought to obtain individual preferences of participants when presented with hedged and non-hedged texts when communicating exams results for hypothetical exams results. This was done across two differing scenarios. The first in a positive context where a hypothetical strong student has obtained a high set of results as shown in Figure 1.

The second in a negative context where a weak student has obtained a low set of exam results. For each of the scenarios the participants are shown the raw exam scores attained and two texts summarising these results: one with hedges and one without as shown in Figures 2 and 3. In total participants were expected to evaluate four different texts. Two for each scenario with one participant judgement expected per scenario.

Whilst the original experiment was conducted with a paper based questionnaire sheet, the repro-

[1] ReproGen - https://reprogen.github.io/

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 282–285,
Aberdeen, Scotland, UK, 20-24 September 2021.



Student Performance I

In the image below you will see the results for a hypothetical student in the United Kingdom in a University. For each course he/she will obtain a CAS score. With **20 being the highest** and **1 being the lowest** obtainable scores. Please read the scenario below carefully.

Imagine a Master's student who has achieved the following exam results:

Course Name	CAS Result
CS5008 - Technologies For The World Wide Web	19
CS5302 - Enterprise Programming	18
CS5401 - Security and Privacy	18
CS5038 - The Electronic Society	18
CS5530 - Strategies for E-Commerce	12
CS5545 - Data Interpretation and Communication	16
CS5544 - E-Technology Workshop	9
CS5553 - Intelligent Architectures	18
CS5942 - MSc Project in E-Technology	9

Figure 1: High exam results table for the first scenario.

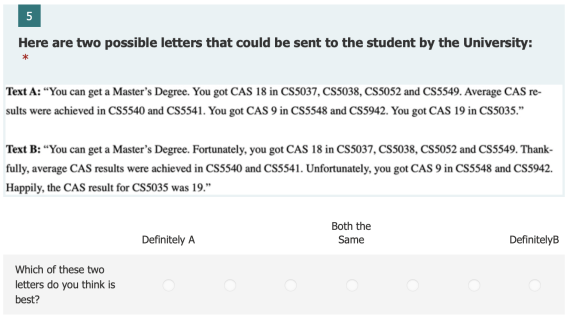
duction used an online based form instead. However, both the questions asked and the format used were mostly identical between the two experiments. The two minor differences being the introduction of additional gender options and the use of age ranges instead of asking participants directly their age.

Participants were asked initially to give their background information. This consisted of their gender (*male*, *female*, *non-binary*, *other*), select an appropriate age-range band, and finally degree of English language proficiency (*Native*, *Non-native, but fluent*, *Not fluent*). Then for both scenarios they were asked to read the results for the student as presented in a table (Figure 1). After this, the participants were asked to state whether they felt the results were good or not for the student (*Yes*, *No*, *Maybe*) and a preference between the two texts A and B. This was done using a Likert scale which ran from -3 for Text A to +3 for Text B. If both texts were considered by a participant to be the same then a score of 0 was given. The participants were asked to provide free text comments on why they made their particular choice of text.

2.2 Participants

The original experiment recruited 37 Masters students (9 females and 28 males). Out of these students only responses from 32 students were used due to incomplete responses from 5 students. From the remaining students 14 participants identified as native English speakers, 11 as non-native but fluent, and 7 as non-fluent English speakers.

Table 1 gives a direct comparison of the participants recruited for the original and reproduction experiments. In contrast the cohort recruited for the



5

Here are two possible letters that could be sent to the student by the University: *

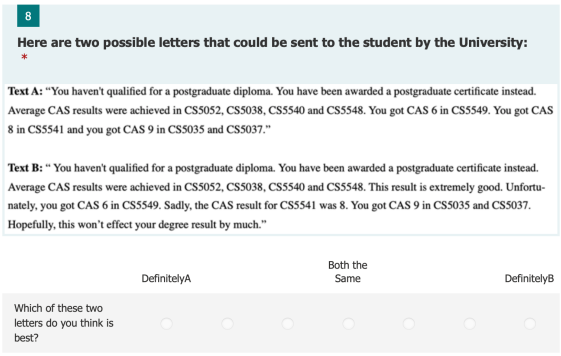
Text A: "You can get a Master's Degree. You got CAS 18 in CS5037, CS5038, CS5052 and CS5549. Average CAS results were achieved in CS5540 and CS5541. You got CAS 9 in CS5548 and CS5942. You got CAS 19 in CS5035."

Text B: "You can get a Master's Degree. Fortunately, you got CAS 18 in CS5037, CS5038, CS5052 and CS5549. Thankfully, average CAS results were achieved in CS5540 and CS5541. Unfortunately, you got CAS 9 in CS5548 and CS5942. Happily, the CAS result for CS5035 was 19."

Definitely A | Both the Same | DefinitelyB

Which of these two letters do you think is best?

Figure 2: Positive student scenario Text A (without hedges) and B (with hedges).



8

Here are two possible letters that could be sent to the student by the University: *

Text A: "You haven't qualified for a postgraduate diploma. You have been awarded a postgraduate certificate instead. Average CAS results were achieved in CS5052, CS5038, CS5540 and CS5548. You got CAS 6 in CS5549. You got CAS 8 in CS5541 and you got CAS 9 in CS5035 and CS5037."

Text B: " You haven't qualified for a postgraduate diploma. You have been awarded a postgraduate certificate instead. Average CAS results were achieved in CS5052, CS5038, CS5540 and CS5548. This result is extremely good. Unfortunately, you got CAS 6 in CS5549. Sadly, the CAS result for CS5541 was 8. You got CAS 9 in CS5035 and CS5037. Hopefully, this won't effect your degree result by much."

DefinitelyA | Both the Same | DefinitelyB

Which of these two letters do you think is best?

Figure 3: Weak student scenario Text A (without hedges) and B (with hedges).

reproduction experiment consisted of colleagues from the author's institution. A total of 11 participants were recruited (4 females and 7 males). Five participants identified themselves as fluent native English speakers and six as non-native but fluent English speakers. No non-fluent English speakers were recruited due to the fact that such participants were not available. Additionally, another key difference between the original experiment and the reproduction is the age of the participants. In the original study 44% (*n=15*) of the participants were under 25 years old, whereas in the reproduction experiment only one participant recruited was in this particular age bracket.

3 Reproduction Results

The results from the reproduction experiment along with the original experiment results for the native and fluent English speaker groups are shown in Table 2. Since there were no non-fluent English speakers recruited results for only native and non-fluent speaker groups are shown. The biggest difference between from the original and reproduction experiments is the results for fluent speakers of English. In the original study this group had shown a

	Native Speakers	Non-Native, but Fluent	Non-Fluent	Total
Original Study	14 (Male: 11, Female: 3)	11(Male: 9, Female: 2)	7 (Male: 3, Female: 4)	32
Repro. Study	6 (Male: 5, Female: 1)	5 (Male: 2, Female: 3)	0	11

Table 1: Comparison of participant numbers between the original and reproduction studies.

weak preference for hedge texts on average, However, in the reproduction this group like the native speakers show an overall strong preference for non-hedged texts in both scenarios. This difference could potentially be explained by difference in the type of participants (Master students vs. working professionals) recruited between the two studies.

For native speakers, the results of the reproduction confirm the initial findings that native speakers prefer the non-hedged over the hedged texts. Interestingly, like the original study native speakers tend to prefer the non-hedged texts to a higher degree than compared to fluent speakers. Although this effect is less pronounced than compared to the original study.

A two-sample T-test was conducted to compare the mean rating score of the native and fluent speaker groups for both scenarios[2]. For the first scenario the result was $t(9)=-0.301$, $p=0.769$ and for the second scenario it was $t(9)=-0.056$, $p=0.956$. The statistically non-significant p-values for both scenarios indicate that the mean rating scores given by both groups for each scenario are not statistically different from each other.

Analysis of free-text comments from fluent speakers across both scenarios showed that participants found the hedges "didn't add value" and that the non-hedged texts were more "formal" and "professional". These comments align with the general comments from native speakers from the original study. It is possible that the use of fluent speakers with professional experience of using English results in cultural expectations that are closer to that of native speakers than compared to fluent speaking students of the original study. Therefore the need for hedges to act as "emotional navigators" are significantly diminished for non-native fluent speakers.

4 Conclusion

In this paper we have conducted a reproduction of a previous NLG study. Unfortunately, we have only been able to only partially replicate the results from the original study. Whilst, we were able to confirm the findings for native speakers we were not able to do so for fluent speakers. This suggest two things. Firstly, that reproduction is a necessary step to better understand the validity of results obtained in initial experiments. And until those results have been validated by a reproduction effort such results should be taken with a degree of scepticism. The second key point is that results obtained in earlier studies cannot be generalised beyond a particular target group of human participants until a reproduction effort confirms the same effect with a different audience. In the case of this study, the original experiment was conducted with Master students. It is possible the effects found maybe limited to that audience in particular. Therefore, it is critical that key demographic information is recorded in human evaluiations to enable future reproduction efforts to have the correct participant mix for their experiments.

	Native	Fluent
Original: S1	-1.42 (σ 2.39)	0.09 (σ 2.59)
Original: S2	-2.07 (σ 1.25)	0.45 (σ 2.53)
Repro: S1	-2.2 (σ 1.09)	-2.0 (σ 1.09)
Repro: S2	-1.40 (σ 2.07)	-1.33 (σ 1.86)

Table 2: Results from the original and reproduction studies for native and fluent speakers. S1 or S2 refers to a particular scenario.

The two key limitations of this reproduction effort is the differences in participant types and the lack of non-fluent English speakers recruited for the study. Therefore, due to the second limitation in particular, it was not possible to confirm or reject a key claim from the previous study that non-fluent speakers prefer texts that contain hedge phrases. This remains an area open for a possible future follow-up reproduction effort.

[2]Reproduction experiment data and analysis code - https://github.com/Saad-Mahamood/reprohum2021

References

Anya Belz, Shubham Agarwal, Anastasia Shimorina, and Ehud Reiter. 2020. ReproGen: Proposal for a shared task on reproducibility of human evaluations in NLG. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages

232–236, Dublin, Ireland. Association for Computational Linguistics.

Anya Belz, Shubham Agarwal, Anastasia Shimorina, and Ehud Reiter. 2021. A systematic review of reproducibility research in natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 381–393, Online. Association for Computational Linguistics.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Saad Mahamood, Ehud Reiter, and Chris Mellish. 2007. A comparison of hedged and non-hedged NLG texts. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*, pages 155–158, Saarbrücken, Germany. DFKI GmbH.

Another PASS: A Reproduction Study of the Human Evaluation of a Football Report Generation System

Simon Mille
Universitat Pompeu Fabra
Barcelona, Spain
simon.mille@upf.edu

Thiago Castro Ferreira
Federal University of Minas Gerais
Belo Horizonte, Brazil
thiagocf05@ufmg.br

Anya Belz and **Brian Davis**
ADAPT Research Centre
Dublin City University
Dublin 9, Ireland
{anya.belz,brian.davis}@adaptcentre.ie

Abstract

This paper reports results from a reproduction study in which we repeated the human evaluation of the PASS Dutch-language football report generation system (van der Lee et al., 2017). The work was carried out as part of the ReproGen Shared Task on Reproducibility of Human Evaluations in NLG, in Track A (Paper 1). We aimed to repeat the original study exactly, with the main difference that a different set of evaluators was used. We describe the study design, present the results from the original and the reproduction study, and then compare and analyse the differences between the two sets of results. For the two 'headline' results of average Fluency and Clarity, we find that in both studies, the system was rated more highly for Clarity than for Fluency, and Clarity had higher standard deviation. Clarity and Fluency ratings were higher, and their standard deviations lower, in the reproduction study than in the original study by substantial margins. Clarity had a higher degree of reproducibility than Fluency, as measured by the coefficient of variation. Data and code are publicly available.[1]

1 Introduction

Recent years have seen growing interest in, and concern about, reproducibility across the Natural Language Processing (NLP) field. The ReproGen Shared Task on Reproducibility of Human Evaluations in Natural Language Generation (Belz et al., 2020a) was the first shared task to focus on reproducibility of human evaluations (rather than metrics). We report on our participation in ReproGen, where our contribution was in Track A, the Main Reproducibility Track. More specifically, we repeated the human evaluation study reported by van der Lee et al. (2017). In this paper, we describe how we approached this task, present the results obtained, and compare our results with those reported in the original paper, using different methods of analysis.

2 Summary of the Evaluated System

PASS (Personalized Automated Soccer texts System) is a modular data-to-text system that produces Dutch summaries of football matches and is a partial re-implementation of the GoalGetter system (Theune et al., 2001). Like GoalGetter, PASS is a template and rule-based system. Unlike GoalGetter, PASS (i) tailors the tone of football reports for supporters of one of the clubs in a match, (ii) has a modular architecture, and (iii) uses templates informed by the MEmo FC (Multilingual Emotional Football Corpus) corpus (Braun et al., 2016).

Data and Language Sources: Automatically scraped football match data from Goal.com,[2] subsequently stored in XML-format, is used as input data, and the MEmo FC corpus as reference data.

System Architecture: The PASS[3] architecture is a data-to-text pipeline consisting of the following modules: (1) the ***governing module*** (used in slightly different versions for different report parts) processes topics one by one, and interacts with the other modules as necessary; (2) the ***topic collection module*** extracts topics from the match data and orders them; (3) the ***lookup module*** retrieves all matching template categories for a given match event and their corresponding templates from a

[1] https://github.com/ThiagoCF05/ReproGen2021-vanderLee

[2] https://www.goal.com/

[3] https://github.com/TallChris91/PASS

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 286–292,
Aberdeen, Scotland, UK, 20-24 September 2021.

database; (4) the ***between-text variety module*** removes templates that were used in the last match report to ensure variety; (5) the ***ruleset module*** checks whether constraints associated with a given template category are met; (6) the ***template selection module*** selects templates from the remaining categories in a weighted random fashion; (7) the ***template filler module*** fills empty template slots with the relevant information from the match data; (8) the ***text collection module*** combines the text produced for the different report parts in the right order; (9) the ***information variety module*** removes repeated information; and (10) the ***reference variety module*** replaces repeated referring expressions.

3 Study Design

We aimed to keep all aspects of study design the same to the extent that was possible. In the sections below, we consider different aspects of study design and describe common features and differences, before summarising same/different properties in Section 3.5.

3.1 Evaluated Texts

The evaluations used ten pairs of alternative system outputs randomly selected[4] from the reports for all football matches from one season of one Dutch league (see top of Figure 1 for an example pair). In each pair, both reports are generated by PASS for the same match, but one report is tailored for supporters of one team, the other for supporters of the other team. Each pair of reports was evaluated by each of the 20 participants.

The questionnaires presented pairs of match reports to evaluators side by side (see Figure 1). Both the order of matches and of report variants for each match was identical in the original and the reproduction study. Sides are not randomised: the report on the left is always for the team in the top answer of the first question in the questionnaire, and the report on the right is always for the team in the bottom answer.[4] This may have made it easier for participants to guess the intended readership, hence contributed to the very high stance identification rates in Table 1.

3.2 Evaluation Criteria

Evaluators were first asked to identify the stance of each text, by completing the statement *Deze tekst is bedoeld voor fans van* ('this text is intended for fans of').[5] Then the quality of the texts was evaluated according to two main criteria, namely *Fluency* and *Clarity*, each of which was assessed via (dis)agreement with two statements: (S1) *Deze tekst is in correct Nederlands geschreven* ('This text is written in correct Dutch' and (S2) *Deze tekst is gemakkelijk leesbaar* ('This text is easy to read') in the case of Fluency; and (S3) *De boodschap van deze tekst is mij geheel duidelijk* ('The message of this text is very clear to me') and (S4) *Tijdens van het lezen van deze tekst begreep ik meteen wat er stond* ('While reading this text, I immediately understood what it said') in the case of Clarity. We used the same Dutch statements as the original study.

All four statements ask the evaluators to consider the text in its own right, that is, texts are not evaluated relative to inputs or an external frame of reference. In terms of the quality criteria properties proposed by Belz et al. (2020b), S3 and S4 have the same properties, whereas S1 and S2 do not. S1 falls into the *Correctness* category (i.e. it is possible to define conditions under which the quality criteria are maximally good), while S2 is in the *Goodness* category (i.e. it is not possible to define such conditions). Another difference between S1 and S2 is that the former considers the form of the text only (independently of the meaning), and the latter takes into account both the form and the meaning.

S3 and S4 have the same basic properties as S2, the three mapping to the specific quality criteria of Understandability, Clarity and Readability, respectively, according to the taxonomy proposed by Howcroft et al. (2020, Appendix D) which incorporates the properties from Belz et al. (2020b) as the top three levels of the taxonomy. S1 maps to Grammaticality. In the taxonomy, Clarity (understandability without effort) is a sub-criterion of Understandability (irrespective of effort), a detail which we return to in the results section (Section 4).

3.3 Evaluation Questionnaire

In the original study, pairs of alternative match reports were presented to evaluators side by side, on the same single page as the evaluation questions (a copy of a page from the original questionnaire is shown on the left of Figure 1). The introduction and ten text pairs were given to evaluators printed out

[4] Information provided via email by the authors of the original paper.

[5] Questions and all other text in the questionnaires were in Dutch. We have provided our own translations.

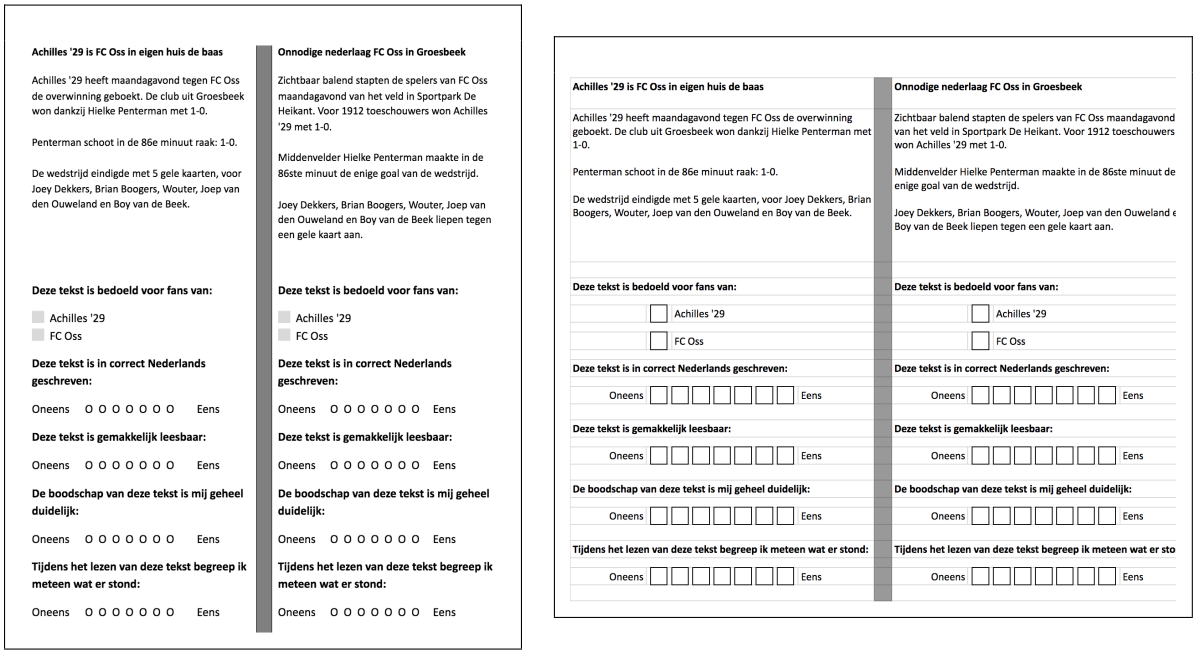


Figure 1: Sample evaluation page from questionnaire in original (left) vs. reproduction study.

on paper. We were unable to do this due to COVID-19 pandemic restrictions, and used online electronic forms instead. The side-by-side text presentation meant we could not use the more commonly used online survey platforms. We opted for a Google Sheet (shown on the right of Figure 1), where only the checkboxes were editable, which made the side-by-side presentation of text pairs possible.

For each of the four quality statements, answers were collected on a 7-point Likert scale (lowest agreement rating 1, highest 7), where the lower (left-hand) side was labeled *Oneens* ('Disagree'), and the higher (right-hand) side *Eens* ('Agree').

Our version of the questionnaire is not identical in every respect, most notably the checkboxes are squares rather than circles, and the alignments and text distribution are slightly different. It cannot entirely be ruled out that such differences affect results, but it seems unlikely.

The recruited participants were provided with two short sets of instructions: (i) the original (Dutch) rating instructions used by van der Lee et al. (2017), in the first tab of the evaluation spreadsheet (updated only to correctly reflect the researchers and institutions involved in the reproduction study), and (ii) additional, specific instructions (in English) relating to the use of the spreadsheet format, in an email that also contained a link to the form.[6]

3.4 Evaluators

In the original study, participants were "all recruited on the campus of the Radboud Universiteit (Nijmegen, The Netherlands). More specifically, [the first author] recruited all participants in the Huygensgebouw of the university, where the faculty of natural sciences, mathematics, and information science is located."[7] Role (student, staff, etc.) and subject area of evaluators was not recorded, but the authors deem it likely that they were students/staff in the faculty subjects (natural sciences, maths, information science) as the faculty's Huygensgebouw building is somewhat isolated on the campus.

We recruited our evaluators remotely (due to COVID-19 pandemic restrictions) via Dutch university research groups known to us, and additionally via personal connections to current and former

[6]Message sent to the participants: "Thank you for accepting to take part in the PASS system evaluation experiment! Below you will find a link to your spreadsheet where your responses will be collected. The sheet contains 12 tabs; we kindly ask you to read carefully the Intro tab and then answer the questions in the following 11 tabs (checkboxes for Page 1 to Page 10, checkbox and free text for Closing). Important notes: (i) The sheet is assigned to you only, and none of the checkboxes or other answers should be filled in when you open it. In the unlikely event that a participant already edited the sheet, please contact us so we can assign you another sheet; (ii) For Pages 1 to 10, please only use lowercase "x" in the checkboxes; you are expected to fill in exactly 10 checkboxes per Page (5 for each text, corresponding to the 5 questions); (iii) Additional instructions are provided (in Dutch) in the Intro tab; (iv) Please complete the evaluation by [DATE]."

[7]Correspondence with the first author of van der Lee et al. (2017).

	van der Lee et al. (2017)	this paper	% in/decrease	CV*
% correctly identified stance	91%	96.75%	+6.32%	6.107
χ^2 for stance identification	233.33†	349.77	–	–
p for χ^2	< 0.001	< 0.00001	–	–
mean Clarity	**5.64**	**6.30**	+11.17%	13.193
stdev	0.88	0.627	-28.75%	–
mean Fluency	**5.36**	**6.14**	+14.18%	16.372
stdev	0.79	0.616	-22.03%	–

Table 1: The results reported in the original paper, alongside the corresponding numbers from our reproduction study. χ^2 is calculated on the contingency table for guessed vs. actual intended stance. CV* is calculated on scores on shifted scales (see in text). † χ^2 is affected by missing values in the original questionnaires.

students and staff in the natural sciences and computer science. This did give us a different cohort of evaluators (e.g. higher average age of 36.8, vs. 20.6 in the original study; some evaluators known to us) and this may be one of the contributing factors to differences in results.

As in the original study, evaluators were not paid or compensated in any other way, and we did not control for demographic balance.

3.5 Summary of Recorded Study Properties

In order to assess reproducibility, and more particularly to be able to compare the degree of reproducibility of different sets of studies, it is important to capture in exactly which respects (in terms of which properties) the reproduction study differs from the original study (Belz, 2021). Below we list the properties in terms of which we *know* whether our reproduction study and the original by van der Lee et al. (2017) were either different or the same, using the basic starter set of properties from Belz (2021), in turn based on Howcroft et al. (2020) and Belz et al. (2020b) (note that system properties don't apply, because the same set of outputs is reused in the present context, rather than regenerated from same inputs):

1. Name and definition of measurand (quality criterion): same.
2. Evaluation modes: same.
3. Method of response elicitation: same.
4. Method for aggregating or otherwise processing raw participant responses: same.
5. Code used to compute and analyse results: different (but only very basic measures were calculated, such as mean and standard deviation).
6. Test set: same.
7. Any preparatory steps such as preprocessing of text taken: same.
8. Procedure of applying measurement method: same.
9. Response collection method: different (paper form in original study, online form in reproduction study, slightly different layout).
10. Quality assurance method(s): different (none in original study which has missing values; checking for completeness and removing questionnaires with all same values in reproduction study).[8]
11. Instructions to evaluators: in evaluation form same, in email different (see Section 3.3).
12. Evaluation interface: different, see Figure 1.

4 Results from Original and Reproduction Study

As described in Section 3.2, the questionnaire contained five rating statements for each text (which we briefly gloss here as 'intended readership', 'correct Dutch', 'easily readable', 'message clear', and 'understood while reading'), but van der Lee et al. (2017) report three scores, for (i) 'intended readership,' (ii) 'correct Dutch' and 'easily readable' combined into a single Fluency score, and (iii) 'message clear' and 'understood while reading' combined

[8] Information provided in direct communication by the authors of the original paper.

		Original study	Reproduction study	CV*
Clarity	S3 avg	5.75 (0.915)	6.36 (0.563)	12.031
	S4 avg	5.52 (0.906)	6.23 (0.686)	14.605
	Both	**5.64**	**6.2975**	13.193
Fluency	S1 avg	5.34 (0.798)	6.22 (0.564)	18.303
	S2 avg	5.41 (0.864)	6.06 (0.661)	13.711
	Both	**5.36**†	**6.14**	16.372

Table 2: Mean scores for the four separate rating statements (Si = ith statement in the questionnaire). Standard deviation in brackets (not corrected for small sample size). CV$*$ calculated on scores on shifted scales (see in text). In our reproduction study, all pairwise differences between S1, S2, S3, S4 are statistically significant at $\alpha = 0.01$ according to a 2-tailed paired t-test, except for the difference between S1 and S4. S1, S2, S3, S4 are also all positively correlated with each other, Pearson's r ranging from 0.36 for S1/S4 to 0.74 for S3/S4. † the mismatch in the average for 'Both' is due to missing values in the original evaluation.

into one Clarity score. The final scores for Fluency and Clarity were calculated by averaging all scores for all texts (both statements and both stance variants) in each case.

We collected 21 evaluations in total, one of which was excluded because the ratings for all questions and all texts were exactly identical in it, which we interpreted as a misunderstanding of the task.[9]

Table 1 shows all results and statistics reported by van der Lee et al. (2017) in Column 2, and the corresponding figures from our reproduction study in Column 3. We also show percentage increases/decreases from original study to reproduction study (Column 4), and the de-biased coefficient of variation (CV*) where appropriate (Column 5), following Belz (2021). The coefficient of variation is the standard deviation over the mean, and is a standard measure of precision used in metrological studies to capture degree of reproducibility. In the implementation we used (Belz, 2021), it is corrected for small sample size. CV* is our primary measure for quantifying the reproducibility of the evaluation scores reported by van der Lee et al. (2017) (stance identification accuracy, mean Fluency and mean Clarity). Note that we shifted all evaluation scales (originally 1..7) to 0..6 prior to computing percentage change and CV*, for fair comparison with the other ReproGen reproduction studies.[10]

As can be seen from the table, all three main evaluation scores went up in our reproduction study: intended stance was correctly identified in 96.75% of cases (compared to 91% in the original study); mean Clarity was 6.3 (compared to 5.64); and mean Fluency was 6.14 (compared to 5.36). Standard deviation for both mean Fluency and mean Clarity went down (better), and the chi-squared value for stance identification and its significance both increased (better).

CV* for Fluency was 16.372 for a mean of 4.75, unbiased sample standard deviation of 0.691 with 95% CI (-3.263, 4.645), and sample size 2. CV* for Clarity was 13.193, for a mean of 4.969, unbiased sample standard deviation of 0.583 with 95% CI (-2.7502, 3.916), and sample size 2. See Belz (2021) for full explanation of this way of reporting CV.

Confidence intervals for (unbiassed) standard deviation (the enumerator in CV*) are large because of the small sample size and corrections incorporated for it. Larger sample sizes increase confidence that the CV for the sample accurately reflects the CV in the general population, and it is important to be clear about level of confidence.

The main conclusions we can draw from the CV* figures is that (i) stance identification is very similar in the two studies, and that (ii) Clarity has a better degree of reproducibility than Fluency.

As mentioned in Section 3.2, according to the standardised quality criteria proposed by Howcroft et al. (2020), S4 is Clarity (understandability without effort), and S3 is Understandability (irrespective of effort); it so happens that Clarity is a subcriterion of Understandability. There are no such parent-child relations between other pairs of S1, S2, S3 and S4, and the taxonomy makes no predictions whether scores for them will be higher or

[9] If we included all 21 evaluations, the average Fluency and Clarity scores would be slightly higher, and degree of reproducibility (CV*) slightly worse.

[10] Both % change and CV in general underestimate variation for scales with a lower end greater than 0.

lower, relative to each other, in the same evaluation. However, the taxonomy does predict that S4 scores (Clarity, or 'understandability without effort') will be lower than S3 scores (Understandability, or 'understandability irrespective of effort') in the same evaluation, because a text that is understandable with effort is also understandable irrespective of effort, but not vice versa. In Table 2, the average S3 score is 6.36, while the average S4 score is indeed lower, at 6.23. This was also the case in the original evaluation where average S3 = 5.752 and average S4 = 5.518. The differences in question were statistically significant at $\alpha = 0.01$ in our reproduction study (see also Table 2).

The taxonomy also predicts that reproducibility (CV^*) and standard deviation will be worse for S4 than S3, which again is borne out in the case of both original and reproduction evaluation by the figures in Table 2. Regarding the other CV^* figures in the last column of Table 2, this is highest (worst) by some margin for S1 ('Grammaticality'), and lowest (best) for S3 ('Understandability'), closely followed by S2 ('Readability') and S4 ('Clarity'). This may come as a surprise as it might be expected that Correctness-type evaluation measures (such as S1) are more reproducible than Goodness-type evaluation measures (S2, S3, S4), which on the face of it involve less clear-cut judgments (e.g. there are no maximally good outputs).

5 Conclusion

In this paper, we reported work which aimed to repeat the human evaluation experiment reported by van der Lee et al. (2017) as closely as possible. We characterised the properties which we know to be either the same or different in our reproduction study (compared to the original study), and presented scores from the reproduction study side by side with scores reported in the original paper (Table 1). We computed percentage increase/decrease, and coefficient of variation as the measure of degree of reproducibility. We found that on the whole, our reproduction study rated the PASS system more highly than the original, with considerably less variation among raters. Furthermore, stance identification accuracy had the highest degree of reproducibility, and mean Clarity scores had a higher degree of reproducibility than mean Fluency.

Note that we have not speculated about the likely reasons for the differences between the two sets of results. We know that those properties marked as different in Section 3.5 are all *possible* reasons. Out of these, it would seem likely that a sizeable part of the difference is down to the different cohorts of evaluators: older, known to us, mostly from computer science backgrounds in the reproduction study, vs. younger, random passers-by recruited in the science building of a university in the original study.

The human evaluation studied here is about as simple as such evaluations get: just one system was evaluated, on three quality criteria and 10 output pairs, each evaluated by the same 20 raters. The coefficient of variation gives a measure of degree of reproducibility that is comparable across measures and across studies, so we can e.g. make the (relative) assessment that Clarity was found to have a higher degree of reproducibility than Fluency. However, the measure does not enable us to make an (absolute) assessment whether either one of them had *good* reproducibility. In order to do this, we would have to know what normally counts as good reproducibility in similar circumstances in NLP. Since NLP currently has very few reproduction studies, and none that report coefficients of variation for human evaluations, such assessments are not possible at this point in time. They will become possible over time if more studies start to report CV (or other measures of precision) for reproduction studies.

Acknowledgments

Mille's work on this study was supported by the European Commission under the H2020 program contract numbers 786731, 825079, 870930 and 952133, and Castro Ferreira's by the Brazilian agency CAPES under Post-doctoral grant No. 88887.508597/2020-00.

References

Anya Belz. 2021. Quantifying reproducibility in NLP and ML. *arXiv preprint arXiv:2109.01211*.

Anya Belz, Shubham Agarwal, Anastasia Shimorina, and Ehud Reiter. 2020a. ReproGen: Proposal for a shared task on reproducibility of human evaluations in NLG. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 232–236, Dublin, Ireland. Association for Computational Linguistics.

Anya Belz, Simon Mille, and David M. Howcroft. 2020b. Disentangling the properties of human evaluation methods: A classification system to support

comparability, meta-evaluation and reproducibility testing. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 183–194, Dublin, Ireland. Association for Computational Linguistics.

Nadine Braun, Martijn Goudbeek, and Emiel Krahmer. 2016. The multilingual affective soccer corpus (MASC): Compiling a biased parallel corpus on soccer reportage in English, German and Dutch. In *Proceedings of the 9th International Natural Language Generation conference*, pages 74–78, Edinburgh, UK. Association for Computational Linguistics.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Chris van der Lee, Emiel Krahmer, and Sander Wubben. 2017. PASS: A Dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 95–104, Santiago de Compostela, Spain. Association for Computational Linguistics.

M. Theune, E. Klabbers, J. R. De Pijper, E. Krahmer, and J. Odijk. 2001. From data to speech: a general approach. *Natural Language Engineering*, 7(1):47–86.

A Reproduction Study of an Annotation-based Human Evaluation of MT Outputs

Maja Popović and **Anya Belz**
ADAPT Centre
School of Computing
Dublin City University, Ireland
{maja.popovic,anya.belz}@adaptcentre.ie

Abstract

In this paper we report our reproduction study of the Croatian part of an annotation-based human evaluation of machine-translated user reviews (Popović, 2020). The work was carried out as part of the ReproGen Shared Task on Reproducibility of Human Evaluation in NLG. Our aim was to repeat the original study exactly, except for using a different set of evaluators. We describe the experimental design, characterise differences between original and reproduction study, and present the results from each study, along with analysis of the similarity between them. For the six main evaluation results of Major/Minor/All Comprehension error rates and Major/Minor/All Adequacy error rates, we find that (i) 4/6 system rankings are the same in both studies, (ii) the relative differences between systems are replicated well for Major Comprehension and Adequacy (Pearson's > 0.9), but not for the corresponding Minor error rates (Pearson's 0.36 for Adequacy, 0.67 for Comprehension), and (iii) the individual system scores for both types of Minor error rates had a higher degree of reproducibility than the corresponding Major error rates. We also examine inter-annotator agreement and compare the annotations obtained in the original and reproduction studies.

1 Introduction

Interest in, and concern about, reproducibility is growing in Natural Language Processing (NLP). Reproducibility of human evaluations, however, has received next to no attention, and the ReproGen Shared Task[1] on Reproducibility of Human Evaluations in Natural Language Generation (NLG) addresses this lack. We participated in ReproGen with a contribution in Track B, the Reproduce Your Own Track. More specifically, we repeated the human evaluation of a mixed set of movie and product review translations produced by three leading Machine Translation (MT) systems, as reported by Popović (2020). In this paper, we summarise the original study in terms of the overall evaluation method (Section 2.1), the quality criteria underlying the annotations from which evaluation scores were derived (Section 2.2), the annotation process and instructions (Section 2.3), and the process by which reviews were selected and translated for the evaluation (Section 2.4). We then present Comprehension and Adequacy error rate results from the original and reproduction studies side by side, and look at how similar system rankings and individual scores are in the two studies (Section 3). Next we compare the inter-annotator agreement in the two studies (Section 4) using diverse metrics. We discuss and interpret results obtained in our reproduction study (Section 5), and draw some conclusions (Section 6).

2 Study Design

2.1 Evaluation Method

The core idea behind the annotation-based evaluation method proposed by Popović (2020) is that instead of assigning overall scores to each sentence,[2] or classifying each error into a predefined error scheme, evaluators mark up word spans in translated texts that contain given types of errors. Two error types, corresponding to the two quality criteria *Comprehensibility* and *Adequacy* (see also Section 2.2), were marked up at two levels of severity (*Major* and *Minor*). The method yields both overall error-rate scores (percentage of words that have been marked up for each error type), and a basis for further quantitative and qualitative analysis of errors and challenging linguistic phenomena. In contrast, current manual evaluation methods for

[1] https://reprogen.github.io/

[2] When we say 'sentence' we mean any sentence-like segment, which may consist of just one word or phrase.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 293–300,
Aberdeen, Scotland, UK, 20-24 September 2021.

MT typically ask annotators either to assign overall per-sentence scores, or to rank two or more translations in terms of given quality criteria, i.e. information about any errors that motivate scores/rankings is not recorded. The method can be applied to any language generation task, genre/domain and language (pair), and can be guided by diverse error types (quality criteria).

2.2 Quality Criteria and Error Rates

The two quality criteria underlying error annotations were *Comprehensibility* and *Adequacy*, both commonly used in MT (ALPAC, 1966; White et al., 1994; Roturier and Bensadoun, 2011).

Comprehensibility: The degree to which a text can be understood. When evaluating Comprehensbility of a translated text, the source language text *is not* shown to evaluators. In terms of the classification system proposed by Belz et al. (2020), Comprehensibility captures the *goodness* of both the *form and content* of a *text in its own right*, and is assessed here by a *subjective, absolute, intrinsic* evaluation measure.

Adequacy (in MT): The degree to which a translation conveys the meaning of the original text in the source language. When evaluating adequacy of a translated text, the source language text *is* shown to evaluators. In terms of Belz et al.'s classification system, Adequacy captures the *correctness* of the *content* of a text *relative to the input*, and is assessed here also by a *subjective, absolute, intrinsic* evaluation measure.

Annotators were asked to mark up translations first for Comprehensibility, then for Adequacy, distinguishing two levels of severity for each: major errors (incomprehensible/not conveying the meaning of the source) and minor errors (difficult to understand due to grammar or stylistic errors/not an optimal translation choice for the given source). Six error rates were then calculated from the mark-up: *Comprehensibility-All*, *Comprehensibility-Major*, *Comprehensibility-Minor*, *Adequacy-All*, *Adequacy-Major*, and *Adequacy-Minor*. These are simply the percentage of words that are part of a text span that has been marked up in the given error category.

2.3 Annotation Process

Annotators first marked up all issues related to Comprehensibility in the translated text without access to the source text. Next, they marked up all issues related to Adequacy while also referring to the source text.

The translated texts were given to the evaluators in the form of a Google Doc, and they were asked to mark major issues with red colour and minor issues with blue colour. In addition to general definitions of Comprehensibility and Adequacy, the evaluators were given detailed guidelines which can be found in the original paper (Popović, 2020).

Evaluators were first given a small number of practice texts to annotate in order to familiarise themselves with the process and clarify any questions and uncertainty. In the original study, these texts were included in calculating the reported results. However, during this practice round in the original study the number and distribution of evaluators varied, which was not repeatable. Therefore, in the reproduction study, the practice texts are not included in calculating reported results, and the results from the original study included in this paper have been adjusted accordingly.

In both studies, each translated review was annotated by two evaluators. All evaluators in both studies were fluent in the source language and native speakers of the target language. However, the backgrounds of the two groups of evaluators are different. In the original study, all seven evaluators working on the Croatian translations were either students or researchers in computational linguistics. Six evaluators had some experience with human translation, and three had experience with machine translation. Three evaluators had a technical background. In contrast, all the evaluators in the reproduction study were translation students, so had the same background and the same or very similar levels of experience with translation.

2.4 Data

The original study involved translations in two similar target languages, Croatian and Serbian, while the reproduction study involved only the Croatian translations, partly for reasons of cost, and partly due to availability of evaluators.

28 English reviews from the Large Movie Review Dataset v1.0[3] (Maas et al., 2011) were selected, as well as 122 English reviews from the 14 categories[4] of the 2018 version of the Amazon

[3] https://ai.stanford.edu/~amaas/data/sentiment

[4] Beauty, Books, CDs and Vinyl, Cell Phones and Accessories, Grocery and Gourmet Food, Health and Personal Care,

	reviews	sentences
	116	894
Amazon MT outputs	68	557
Bing MT outputs	35	279
Google MT outputs	61	467
total MT outputs	164	1303

Table 1: Number of evaluated reviews and sentences.

Product Review dataset[5] (McAuley et al., 2015). In the selection process, overly long ($>$ 350 words) and overly short ($<$ 30 words) reviews were excluded, and an equal number of positive and negative reviews were selected to ensure balance in terms of sentiment polarity, while a balanced distribution between topics in Amazon reviews was also aimed for.

The selected (English) user reviews were then translated into Croatian using Google Translate, Bing and Amazon Translate, yielding a total of 450 Croatian translations of which 164 were arbitrarily selected as a manageable number for inclusion in the evaluation. The 164 selected translations correspond to 116 original English reviews which were mostly translated by one, and in some cases by two, of the MT systems, in order to increase diversity in translations (hence in error types). 68 of the translations were produced by Amazon Translate, 35 by Bing Translator and 61 by Google Translate. The reason for including fewer Bing translations was their notably lower quality. The number of reviews and sentences evaluated for each system can be seen in Table 1.[6]

The primary aim of the original study was not the comparative evaluation of multiple MT systems. Rather, the aim was to test a new evaluation scheme. The system-level error rates reported in the next section can therefore not be considered a fair assessment of the respective quality of the three systems involved. For this purpose, normally the same source texts translated by all systems would be evaluated. In the present context, different source texts translated by different systems were evaluated, chosen as explained above. Nevertheless, assessments of the reproducibility of the obtained human evaluation scores are valid regardless of this diversity in test sets, provided the latter are the same in the original and the reproduction study.

3 Evaluation Scores

Columns 2–7 in Table 2 show the overall evaluation scores obtained in the original and in the reproduction study in the form of error rates, i.e. percentages of words marked up as errors. The following tendencies can be observed in both studies: (i) four out of six system rankings (the exceptions being Major Comprehension and Minor Adequacy) are the same in both studies; (ii) error rates were higher for Comprehension than for Adequacy in all three error subcategories and for all systems, except that the Adequacy-Minor rate for Bing was higher than its Comprehension-Minor rate in the original study which also affected the corresponding All rate; (iii) Bing exhibits the highest error rates in all error categories except Comprehension-Minor and Adequacy-Minor in the reproduction study; and (iv) Google has slightly lower error rates than Amazon in all error categories except for Comprehension-Major and Adequacy-Major in the reproduction study, and Adequacy-Major in the original study).

The last three columns in Table 2 show the coefficient of variation (CV) for each of the individual error-rate scores across the two studies as our primary measure of degree of reproducibility (Belz, 2021). CV is a standard measure of precision in metrological studies of reproducibility.[7] The main general tendencies are as follows: (i) the Adequacy-All and Adequacy-Minor error rates (except for the Minor rate for Bing) have better reproducibility (CV is lower) than the corresponding Comprehension rates; and (ii) the Adequacy-Major error rates (except for the Major rate for Google) have worse reproducibility (CV is higher) than the corresponding Comprehension-Major rates.

Table 3 shows Pearson's r between the system-level Comprehension and Adequacy error rates in the original and the reproduction studies, for each of the All, Major and Minor subcategories. A clear pattern can be observed: correlation between system scores in the Minor categories is far worse than in the All and Major categories.

Since there are only three systems to calculate correlation on, we also calculated Pearson's r between sentence-level error counts and the results are presented in Table 4. The picture confirms the

Home and Kitchen, Movies and TV, Musical Instruments, Patio, Lawn and Garden, Pet Supplies, Sports and Outdoors, Toys and Games, Video Games.

[5] http://jmcauley.ucsd.edu/data/amazon/

[6] The annotated data sets resulting from both the original study and the reproduction study are publicly available under the Creative Commons CC-BY licence here: https://github.com/m-popovic/QRev-annotations

[7] We used the de-biased version of CV, for small samples, as proposed by Belz (2021).

System	Comprehension error rate (%)								
	original study			reproduction study			coefficient of variation (CV)		
	All	Major	Minor	All	Major	Minor	All	Major	Minor
All	21.9	9.2	12.7	29.6	13.4	16.2	29.81	37.06	24.15
Amazon	19.6	**7.6**	12.0	26.9	**10.2**	16.7	31.30	29.13	32.65
Bing	31.1	15.1	16.0	39.1	22.3	16.8	22.72	38.38	4.86
Google	18.3	**7.1**	11.2	26.5	**11.5**	15.0	36.498	47.17	28.92

System	Adequacy error rate (%)								
	original study			reproduction study			coefficient of variation (CV)		
	All	Major	Minor	All	Major	Minor	All	Major	Minor
All	21.1	8.2	12.9	24.8	12.3	12.5	16.07	39.88	3.14
Amazon	17.9	6.5	**11.4**	22.6	9.5	**13.1**	23.14	37.39	13.84
Bing	30.2	13.2	**17.0**	33.9	21.2	**12.7**	11.51	46.37	28.87
Google	17.5	7.0	10.5	21.4	9.7	11.7	19.99	32.24	10.78

Table 2: Error rates (percentages of words that are marked problematic) for Major/Minor Comprehensibility and Adequacy in Croatian translated texts in the two evaluation studies, shown for the three MT systems combined (All) and individually. CV between error rates in original and reproduction for each error category, using the de-biased version of CV proposed by Belz (2021). Bold indicates different system rank in original/reproduction studies.

System-level scores		
	Comprehension	Adequacy
All	0.9979**	0.9982**
Major	0.9882*	0.9986**
Minor	0.6663	0.3623

Table 3: Pearson correlation coefficients between system-level scores in the original and reproduction studies. ** = significant at $\alpha = 0.01$; * = significant at $\alpha = 0.05$.

Sentence-level scores		
	Comprehension	Adequacy
All	0.695**	0.720**
Major	0.580**	0.656**
Minor	0.403**	0.390**

Table 4: Pearson correlation coefficients between original sentence-level error counts in the original and reproduction studies. (All significant at $\alpha = 0.01$.)

system-level correlation results: while All and Major error counts correlate reasonably well for both error types (although slightly better for Adequacy than for Comprehension), the coefficients for the Minor error types are notably lower.

We will return to some of the above points in the discussion section (Section 5).

4 Inter-annotator Agreement

The original study reported inter-annotator agreement (IAA) in terms of F-score and normalised edit distance (definitions below). In this paper we also report Krippendorff's α for both original and reproduction study, following Kreutzer et al. (2020) who used it in a similar error marking study.[8]

Krippendorff's α: In order to quantify agreement by this method, error annotations were converted to a sentence-level quality score, namely the *number* of words marked up for error in a given sentence. For a perfect sentence, no words would be marked so this score would be zero. Using the standard definition,[9] we computed three separate α scores: (i) from just the Major error annotations, (ii) from just the Minor error annotations, and (iii) from both (corresponding to the All subcategory from previous sections).

F-score: To compute sentence-level F1-score, the starting point was the paired sequences *ev1* and *ev2* of word-level error labels (*Major*, *Minor* or *None*) assigned by the two annotators to a sentence. Precision was then computed as the labels from *ev1* also present in *ev2,* and Recall as labels from *ev2* also present in *ev1*. The F1-score was then calculated in the usual way, as the harmonic mean of Precision and Recall. Due to possible length differences in a pair of label sequences (due to insertion of X labels representing missing words), matches are defined as position-independent, which can result in overestimation of agreement.

To yield system-level scores, sentence-level scores are micro-averaged by aggregating matches and lengths.

Edit distance: The standard definition of edit distance[10] with insertions, deletions and substitutions all at cost=1 is applied to paired sequences *ev1* and *ev2* of word-level error labels (as above). Nor-

[8] Cohen's kappa was not considered appropriate for either of the studies for the reasons explained in detail in the original paper (Popović, 2020).

[9] https://en.wikipedia.org/wiki/Krippendorffs_alpha

[10] Also known as Levenshtein distance (Levenshtein, 1966).

malised edit distance scores are obtained by dividing the summed cost of edits by sequence length. However, while usually (in speech recognition and MT), normalisation is carried out by the length of the 'correct' reference string, here neither of the label sequences is (in)correct. Therefore, the edit-distance metric is symmetrised (in a similar way as the F-score is with Precision and Recall) by first computing edit distance of *ev1* against *ev2*, then of *ev2* against *ev1*, then summing over both and normalising by the sum of the lengths of *ev1* and *ev2*. The resulting measure does penalise differences in label position, thus compensating for the drawback of the position-independent F-score above.

To yield system-level scores, sentence-level scores are micro-averaged by aggregating edit distances and lengths.

Illustration of IAA metrics: Examples of two sentences annotated by two different annotators, along with counts obtained in computing the metrics, are shown in Table 5.

The first two rows show the annotated texts as described in Section 2.3, namely major errors in red/bold, and minor errors in blue/italics. The next two rows below show the extracted error label sequences that form the basis for measuring agreement. The abel sequences were used directly for calculating F-score and edit distance, whereas for Krippendorff's α, label counts were derived instead (rows 5, 6 and 7).

IAA scores computed with the above metrics for the original and reproduction studies are shown in Table 6. IAA is generally good in both studies in terms of all metrics. Furthermore, all metrics indicate a higher IAA for Adequacy than for Comprehensibility (although in some cases differences are very small). Another clear tendency for both studies is a notably lower Krippendorff's α for error annotations in the Minor categories than in the Major and All categories.

For Comprehension errors, IAA is better in the original study according to all metrics. For Adequacy errors, F-score, edit distance and α for Minor errors are also better in the original study, while α for All and Major errors are better in the reproduction study.

5 Discussion

In previous sections, we presented results and similarities/differences observable in them in objective terms. In this section, we discuss and interpret results, aiming to draw conclusions and to identify reasons for similarities and differences.

5.1 Differences in overall scores

As mentioned in Section 3, both studies show broadly similar tendencies in error rates, with some exceptions for Major Comprehension errors and Minor Adequacy errors, as follows. In terms of Major Comprehension error rates from the reproduction study, Amazon is slightly better than Google, while the original study indicates the opposite. As for Minor Adequacy errors, the original study clearly indicates that the Bing translations contain the largest number of errors, which is in line with other scores, too. In the reproduction study, annotators found fewer Minor Adequacy errors in Bing translations than in Amazon translations, however the number of Major Adequacy errors for Bing is much higher in the reproduction study than in the original one. Apparently the two groups of annotators perceived similar overall number of errors but different distributions between Major and Minor ones.

Taking into account the lower inter-annotator agreement for Minor errors, as well as the fact that in both studies the majority of evaluators reported that it was often difficult to distinguish between major and minor errors, the difference in Minor Adequacy errors is not very surprising. As for Major Comprehension errors, lower inter-annotator agreement and larger degree of subjectivity in assessing Comprehensibility may contribute to the slight difference in scores.

Both system-level and sentence-level error rates correlated better across the stwo studies for Major error types than for Minor error types. This also points in the direction of minor errors being generally harder to annotate reliably, something that will need to be addressed in future evaluations.

In terms of the pairwise degree of reproducibility captured by CV, individual pairs of Major error rates differed more between the two studies than individual pairs of Minor error rates. This is not a contradiction with other results: CV measures how far off each individual score is from its original counterpart, whereas Pearson's r measures covariance between *sets* of original and reproduction scores, i.e. how similar their relative ranks and the distances between scores are. In other words, in our results, Major error rates are on average further apart in absolute terms, but evince a more similar

text annotated by ev1	**Ne shvaćajte ih ako udarite u tešku torbu .**
text annotated by ev2	Ne **shvaćajte** ih ako udarite u **tešku torbu** .
error labels, ev1	Major Major Major Major Major Major Major Major Major
error labels, ev2	None Major None None None None Major Major None
major error counts, ev1 ev2	9 3
minor error counts, ev1 ev2	0 0
total error counts, ev1 ev2	9 3
F score (matching labels)	33.3 (3 label matches, total number of labels e1 = 9, e2 = 9)
edit (unmatched labels)	66.0 (6 label mismatches)
text annotated by ev1	*Nadmašio* me na svakom koraku i stalno *me iznenadila priča* .
text annotated by ev2	**Nadmašio me X** na svakom koraku i stalno me *X iznenadila* priča .
error labels, ev1	Minor None None None None None None Minor Minor Minor None
error labels, ev2	Major Major Minor None None None None None None Minor Minor None None
major error counts, ev1 ev2	0 2
minor error counts, ev1 ev2	4 3
total error counts, ev1 ev2	4 5
F score (matching labels)	83.3 (10 label matches. total number of labels e1=11, e2=13)
edit distance (unmatched labels)	25.0 (3 label mismatches)

Table 5: Illustration of IAA metrics: two sentences annotated for comprehension by two evaluators, error labels, error counts used for Krippendorff's α, F-score on labels and edit distance on labels. Bold/red stands for major errors, italics/blue for minor errors, and an X represents an omitted word.

IAA	Comprehension					Adequacy				
	$\uparrow \alpha$			$\uparrow$ F	$\downarrow$ edit	$\uparrow \alpha$			$\uparrow$ F	$\downarrow$ edit
	major	minor	all	score	dist.	major	minor	all	score	dist.
original	0.621	0.412	0.687	82.3	22.3	0.679	0.420	0.699	84.1	19.9
reproduction	0.467	0.363	0.636	76.4	30.0	0.734	0.394	0.723	83.0	23.2

Table 6: IAA scores for Comprehensibility and Adequacy: Krippendorff's α, F-score and normalised edit distance.

overall picture in relative terms, than Minor error rates, which does appear to support the conclusion that Minor errors are harder to agree on, and that the dividing line between Major and Minor errors is also hard to agree on.

5.2 Differences in inter-annotator agreement

Some tendencies in IAA are similar in both studies (Section 4). IAA was reasonably good in both studies in terms of all metrics. A contributing factor is likely to be that the annotators were not asked to perform any fine-grained error categorisation. Another clear tendency for both studies was a notably lower Krippendorff's α for minor errors: since these tend to be far less severe (not completely unintelligible, not entirely changing the meaning of the source text), it may be the case that judgments here reflect personal preferences more.

There were also notable differences between the two studies, including IAA being worse in the reproduction study than the original according to 8 out of 10 measures in Table 6. On the face of it, this is not as expected, given the apparently greater homogeneity of the second cohort of evaluators mentioned above. The two cohorts may have other characteristics not accessible to us that would explain the difference.

Moving on to comparing IAA across different error types, the reason for lower Krippendorff's α for Minor errors is probably the generally greater difficulty of agreeing on Minor error annotations mentioned in Section 5.1.

One possible explanation for all metrics indicating a higher IAA for Adequacy than for Comprehensibility is that Adequacy is guided by the original source text while Comprehensibility relies only on the translated text, possibly allowing more space for subjectivity in judgments.

However, to gain a more complete understanding of the above, future work needs to analyse differences in more detail. There are also potential improvements that can be made in the guidelines which could make a difference to IAA measures (for details see the original paper).

5.3 Mark-up Agreement between the Two Studies

To assess the similarity between the annotations produced in the original and reproduction studies, we paired all strings from the original study with all strings from the reproduction study, and then applied the F1 metric as described in Section 4 above, except that this time we used the *word* strings, not the label strings. Table 7 presents an example of a

	original study	reproduction study
annotations	Običnо **ventilator**, ali *neimpresioniran* Običnо **ventilator**, ali *neimpresioniran*	Običnо **ventilator**, ali *neimpresioniran* Običnо **ventilator**, ali **X** neimpresioniran
all errors prec / rec	ventilator neimpresioniran ventilator neimpresioniran prec = 3 matches / 4 words = 75	ventilator neimpresioniran ventilator X rec = 3 matches / 4 words = 75
major errors prec / rec	ventilator ventilator prec = 2 matches / 3 words = 66.7	ventilator ventilator X rec = 2 matches / 2 words = 100
minor errors prec / rec	neimpresioniran neimpresioniran prec = 1 match / 1 word = 100	neimpresioniran rec = 1 match / 2 words = 50

Table 7: Illustration of annotation overlap metric: example text annotated twice in the original study (left), and twice in the reproduction study (right). Red/bold = major error, blue/italics = minor error, X = omitted word.

	Comprehension	Adequacy
All errors	56.3	58.0
Major errors	54.2	56.4
Minor errors	39.3	36.6

Table 8: Overlap between words marked up in the two studies in terms of word-string F1 score.

sentence annotated in the two studies, all marked-up words, and the corresponding word-string Precision and Recall scores. The corresponding F1 values are shown in Table 8 for all error cateogories.

The main tendency is that the overlap for Minor errors is notably lower than for Major and All errors, providing further evidence that Minor errors are harder to agree on. As for Comprehension vs. Adequacy errors, overlap in Minor annotations is worse for Adequacy (than Comprehension), but overlap in Major and All annotations is worse for Comprehension, which aligns with results from Section 3 that the system rankings for Major Comprehension and Minor Adequacy were switched between the two studies.

6 Conclusion

In this paper, we reported results from a reproduction study of an annotation-based human evaluation of MT outputs where errors related to comprehensibility and meaning correctness were annotated in texts by marking up word involved in an error. We compared the corresponding Comprehension and Adequacy system-level error rates for the three MT systems assessed in the two studies, distinguishing subcategories All, Major and Minor for each. We found that 4 out of 6 system rankings were the same in both studies, but that the relative differences between systems are not well replicated for both types of Minor error rates (Pearson's 0.36 for Adequacy-Minor, 0.67 for Comprehension-Minor). However, the *individual* system scores for both types of Minor error rate had a higher degree of reproducibility (as measured by the coefficient of correlation, CV), than the corresponding Major error rates. Results also showed that Minor Adequacy and Major Comprehension annotations and system rankings differed more than other error categories.

The reproduction study reported here was a contribution to the ReproGen Shared Task in the 'Reproduce Your Own' Track, and as such we had the benefit of having full access to all resources and information from the original evaluation, a luxury not normally available when conducting a reproduction study of someone else's work. The main difference between properties of the original study and our reproduction was the characteristics of the cohort of evaluators who had slightly different backgrounds. There were pronounced similarities between the two studies, but also very clear differences, notably including in system rankings. All in all, while repeating the study was simply a matter of recruiting a new cohort of evaluators, obtaining the same results proved somewhat less simple.

Acknowledgements

Both authors benefit from being members of the ADAPT SFI Centre for Digital Media Technology which is funded by Science Foundation Ireland through the SFI Research Centres Programme, and co-funded under the European Regional Development Fund (ERDF) through Grant 13/RC/2106.

The original study (Popović, 2020) was partly funded by the European Association for Machine Translation (EAMT).

The reproduction study was funded by the ADAPT NLG research group.

References

ALPAC. 1966. Language and machines. Computers in translation and linguistics.

Anya Belz. 2021. Quantifying reproducibility in NLP and ML. *arXiv preprint arXiv:2109.01211*.

Anya Belz, Simon Mille, and David M. Howcroft. 2020. Disentangling the properties of human evaluation methods: A classification system to support comparability, meta-evaluation and reproducibility testing. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 183–194, Dublin, Ireland. Association for Computational Linguistics.

Julia Kreutzer, Nathaniel Berger, and Stefan Riezler. 2020. Correct Me If You Can: Learning from Error Corrections and Markings. *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation (EAMT 20)*.

Vladimir Iosifovich Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707–710.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015)*, pages 43–52, Santiago, Chile.

Maja Popović. 2020. Informative manual evaluation of machine translation output. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5059–5069, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Johann Roturier and Anthony Bensadoun. 2011. Evaluation of MT Systems to Translate User Generated Content. In *Proceedings of the MT Summit XIII*, Xiamen, China.

John White, Theresa O'Connell, and Francis O'Mara. 1994. The ARPA MT evaluation methodologies: evolution, lessons, and future approaches. In *Proceedings of the 1994 Conference of Association for Machine Translation in the Americas*, pages 193–205.

TUDA-Reproducibility @ ReproGen: Replicability of Human Evaluation of Text-to-Text and Concept-to-Text Generation

Christian Richter, Yanran Chen, Steffen Eger
Computer Science Department
Technical University of Darmstadt (TUDA)
chrisrichter145@gmail.com, chenyr1996@hotmail.com, eger@aiphes.tu-darmstadt.de

Abstract

This paper describes our contribution to the Shared Task *ReproGen* by Belz et al. (2021), which investigates the reproducibility of human evaluations in the context of Natural Language Generation. We selected the paper "Generation of Company descriptions using concept-to-text and text-to-text deep models: data set collection and systems evaluation" (Qader et al., 2018) and aimed to replicate, as closely to the original as possible, the human evaluation and the subsequent comparison between the human judgements and the automatic evaluation metrics. Here, we first outline the text generation task of the paper of Qader et al. (2018). Then, we document how we approached our replication of the paper's human evaluation. We also discuss the difficulties we encountered and which information was missing. Our replication has medium to strong correlation (0.66 Spearman overall) with the original results of Qader et al. (2018), but due to the missing information about how Qader et al. (2018) compared the human judgements with the metric scores, we have refrained from reproducing this comparison.

1 Introduction

Reproducibility is an utmost priority in research to ensure reliability of scientific findings. Informally, it describes the ability to repeat a study, beginning with the same starting point, using the same resources (if possible) and achieving the same results and conclusions (Pineau et al., 2020). Reproducibility requires that approaches in publications be recorded in such a way that previously uninvolved parties can comprehend and recreate them (Fokkens et al., 2013). However, reproducibility is a complex requirement which often fails because of missing details (like not described data sets or missing key parameters)—such aspects, even though they may appear minor at first sight, either prevent reproducibility altogether or at least distort the results (Raff, 2019; Wieling et al., 2018). One reason for such failures of reproducibility may be lack of widely accepted definitions and practical conceptualization of reproducibility, as there is currently no consensus on how and to what level of detail research should be documented (Cohen et al., 2018).

The Shared Task *ReproGen* (Belz et al., 2021) deals with the reproducibility problem. In particular, it aims to investigate reproducibility of human evaluation. The findings of ReproGen should yield general insights into how reproducibility can be improved. The task in ReproGen is to replicate either one of the pre-selected studies or a self-selected study from the field of Natural Language Generation (NLG) and to document the findings.

In this paper, we report on our reproducibility of the work "Generation of Company descriptions using concept-to-text and text-to-text deep models: dataset collection and systems evaluation" (`CompDesc` for short) by Qader et al. (2018). This work analyzes multiple sequence-to-sequence models that were used to generate short company descriptions from Wikipedia articles. This includes both automatic and human evaluation which are then compared with each other. Our replication focuses on the human evaluation, in accordance with the general outline of ReproGen.

2 `CompDesc` and our replication

We first describe the paper `CompDesc`, then outline how we replicated its human evaluation. Finally, we compare both experiments.

2.1 The paper `CompDesc`

The paper `CompDesc` first creates a data set of Wikipedia articles about companies[1]. Then, using four concept-to-text and two text-to-text approaches, they generate short summaries out of

[1] https://gricad-gitlab.univ-grenoble-alpes.fr/getalp/wikipediacompanycorpus

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 301–307,
Aberdeen, Scotland, UK, 20-24 September 2021.

this data. Figure 1 shows an example from our experiment, which is what the evaluators can see during the evaluation. The title and the description at the top as well as the info box at the right margin, which are typically present in every Wikipedia article, serve as input. The language generation models then generate the summary either from the description or the info box, depending on the type of the text generation system. Afterwards, Qader et al. (2018) evaluated the system performance on the test set of their Wikipedia company corpus using five automatic evaluation metrics. Table 7 in Appendix A.3 shows the results of the automated evaluation. In addition to that, they conduct a human evaluation using a selection of randomly sampled summaries with 19 test persons where each one evaluated 10 summaries. But the human evaluators did not know that some of the summaries were actually human generated, namely the references. For that, the humans assessed the criteria *information coverage*, *information redundancy*, *semantic adequacy* and *grammatical correctness* on a 5-point Likert scale. Finally, Qader et al. (2018) compared the results of the two evaluation methods.

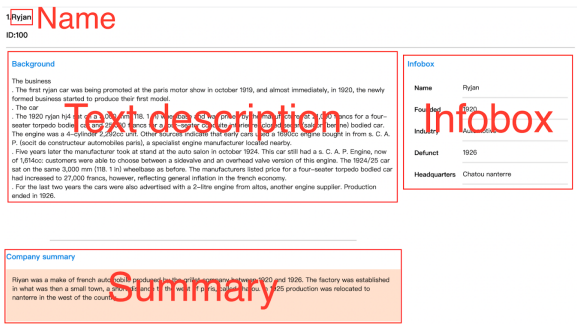


Figure 1: Example of Human Evaluation: the summary is created from the other fields. All 4 boxes are presented during the human evaluation process.

2.2 Replication of `CompDesc`

There were two phases in our replication study: 1) Preparation, where the goals of reproduction and needed resources were determined; 2) the human evaluation experiment, where we collected the human ratings, that were then compared to the original results.

Preparation In the preparation phase, there were three resources initially provided by Belz et al. (2021) as part of the shared task (see Appendix A.1), namely 1) the original paper (Qader et al., 2018), which describes the implementation as well as the methods and data used; 2) an incomplete human evaluation data sheet filled out by the authors of (Qader et al., 2018), which should also be filled out by the participants of the shared task later; 3) a link to a GitLab repository that contains code for a web-based survey tool called "FlexEval" (Fayet et al., 2020). The original code was not available, also not upon request.

Based on the information and resources available, we first identified which results should be replicated: **the average scores of the human evaluation based on a 5-point Likert scale per system** (see Table 1), and, as a secondary goal, **the comparison of human and automatic evaluation metrics using Spearman's correlation** (see Table 4). Then, we determined the resources needed to reproduce the human evaluation, which include the system outputs and references, the data and ideally the code for computing the correlations. However, none of the above was included in the Shared Task resources. Upon request, the authors provided us with parts of the data, including the summaries they used to conduct the human evaluation (both as CSV and HTML files) and a CSV file containing their human evaluation scores, whose reproduction is the primary goal of this report.

Human Evaluation Experiment In order to keep our reproduction as close as possible to the original in terms of content and appearance, the identical data sets were selected for reproduction using the provided HTML files.

In the beginning, a unique identification number was assigned to each summary to match the results to the corresponding summaries. After that, 19 files, each containing 10 summaries, were randomly created out of the original files. In addition, a survey was created using Google Form[2] to collect the evaluator ratings of the four criteria, each on the basis of a 5-point Likert Scale. 19 people from the authors' social environment volunteered as participants for this study. They were not English native speakers, similar to the participants of `CompDesc`. However, `CompDesc` does not explain why these conditions were chosen. This may not have been intentional, but a result of the composition of the participants. We have decided to take this into account anyway. When conducting the human evaluation, each participant was given one of the 19 HTML files and a link to the Google

[2]https://www.google.de/intl/en/forms/about/

Form via E-Mail or other chat apps (see Appendix A.2).

After obtaining the human ratings, we exported the data using the same format as the original one. We calculated the average scores directly based on this file. But for reproducing the correlation matrix, some further resources were needed. Since the paper is ambiguous about how the results were computed and the corresponding code and data were missing, we tested different calculation approaches to determine the original calculation. Unfortunately, it didn't succeed in the end. We could only reproduce a part of the correlation values on the basis of the original human evaluation results that Qader et al. (2018) provided us. Table 5 presents this special case, which we will describe in detail in Section 3. In the end, we examined the similarities between the original and reproduced results.

2.3 Assessment

Comparing with the original experiment, there are several notable differences. First, the original study used "FlexEval" (Fayet et al., 2020) to conduct the survey, which probably showed the evaluation data and corresponding questions side by side in a web application and the evaluators can answer the questions by scrolling down. In their paper, Qader et al. (2018) only stated that they "set up a web-based experiment" (Qader et al., 2018), but they did not mention what tool they used. However, since the tool is very complex to configure and adequate guidance was not available, we used Google Forms[3] instead. However, we made sure that the participants received the same data presentation.

The use of "FlexEval" only became apparent with additional information from the shared task, as the authors mentioned it in the human evaluation data sheet. But in our survey, the presentation of the data and the input mask were accessible through two separate sources. In contrast to the participants of the original study, who were all members of a lab, the participants of the replication were only selected based on their connection to us.

Besides those distinct differences from the original experiment, we made several assumptions because of the inaccuracies and the missing information found during the preparation, which could influence possible deviations of the results between original and replication. We describe these in the following:

[3]https://www.google.de/intl/en/forms/about/

1) There is an inconsistency in the description of the experiments sets. Qader et al. (2018) stated in their paper that each of the 19 participants evaluated 10 summaries, resulting in a total number of 190. However, it was also stated in the paper that 30 summaries were evaluated for all 7 systems (including reference), which makes a total of 210 summaries. When asked, the authors explained that a random selection was made from the 210 summaries. This agrees with the raw human evaluation results we received on request. Therefore, we relied on the explicit specification of 19 times 10 random summaries.

2) Qader et al. (2018) perform a manual quality checking of the results of the human evaluation, but do not go into detail about the procedure. To be able to guarantee a minimum of quality, we considered an evaluation invalid when the majority of the answers were illogical. This occurred only once, where a participant randomly selected the values 3 and 4 independently of the summary quality. In this case, we passed the task to an additional participant for re-evaluation.

Nevertheless, the replication follows the original in the essential points such as the requirements for evaluators, the number of evaluators, the amount of evaluated items, the identical set of questions, the format of data, and the survey guidelines which prohibit to ask questions during the experiment. Therefore, we conclude that, assuming the same basic conditions, a comparison of the results below is legitimate.

3 Results

Table 1 displays the human evaluation results of Qader et al. (2018), whereas Table 2 shows our replicated results. As one can see, different levels of variation show up between the two experiments. Larger deviations of more than one point can only be seen twice, all other deviations are smaller. These deviations may have been caused by various factors. In general, smaller differences are always possible in stochastic environments. It also cannot be ruled out that the differences may result from minor but recurring discrepancies of the score as well as the participants in the two studies could have rated the results fundamentally differently, but with a simple average deviation of 0.47, which is only 14% off the average value.

In addition, we calculated Spearman's ρ and Pearson's r correlations between the values in the

	cover.	non-redun	semant.	gramm.
Reference	3.1	4.6	3.9	4.2
C2T	2.9	2.9	3.3	3.6
C2T_char	2.3	3.9	2.8	3.0
C2T+pg	2.3	4.5	4.0	4.3
C2T+pg+cv	2.7	3.9	3.6	4.2
T2T+pg	1.8	3.3	2.9	3.7
T2T+pg+cv	2.3	3.8	2.4	3.5

Table 1: ORIGINAL: The original human evaluation results taken from Qader et al. (2018).

	cover.	non-redun	semant.	gramm.
Reference	3.9	4.1	3.9	4.0
C2T	2.5	3.8	2.6	3.2
C2T_char	3.0	**2.8**	3.1	3.5
C2T+pg	2.6	4.2	**2.9**	3.8
C2T+pg+cv	3.0	4.1	3.9	4.1
T2T+pg	2.6	3.5	2.7	4.0
T2T+pg+cv	2.9	4.1	2.8	4.4

Table 2: REPLICATION: The replication results of the human evaluation. Differences of more than 1 are bold.

two tables on each axis. From Table 3, we observe that the reproduced evaluation scores for the systems C2T+pg, C2T+pg, T2T+pg and T2T+pg+cv are highly correlated with the original values, but this may be unreliable due to the small number of input values. Unfortunately, we were not able to compare the scores at the summary-level, because of the missing information about the arrangement in the original experiment. However, if we calculate a single correlation using both methods between all values of both tables, we get a more reliable score. The values of 0.66 respectively 0.7 represent a moderate to strong statistical significant correlation (Taylor, 1990; Schober et al., 2018).

	ρ	r
All	0.66*	0.70*
Reference	0.95	0.82
C2T	0.20	-0.05
C2T_char	0.20	-0.32
C2T+pg	1.0*	0.83
C2T+pg+cv	0.80	0.97*
T2T+pg	1.0*	0.86
T2T+pg+cv	0.60	0.95*
cover.	0.41	0.58
non-redun.	0.64	0.38
semant.	0.36	0.54
gramm.	0.14	0.33

Table 3: Spearman's ρ and Pearson's r correlations between Table 1 and Table 2. Values marked with * show a significant correlation ($p \leq .05$).

To figure out how Qader et al. (2018) computed the correlations specifically, we conducted some further experiments based on the original data from Qader et al. (2018), which contains the human judgements for each summary. In the first step, we proved the validity of the data by a successful reproduction of the values in Table 1. Afterwards, we made several attempts regarding the source of the metric scores, the level at which the correlations were computed and whether the correlated values included the scores for the references, to achieve a valid reproduction of the original correlation matrix, using only the original data.

After that, despite not discovering the original setup, there is one noteworthy case (see Table 5) where we successfully reproduced the correlations between the results of the 5 automatic evaluation metrics and that between the human judgements of the 4 criteria (values outside the black square). Surprisingly, a large gap still exists for the comparison between the metric scores and the human scores (values in the black square). We can draw a completely different conclusion from these reproduced correlations. E.g., Table 5 shows that METEOR, ROUGE-L, and CIDEr are highly correlated with redundancy (green marker), but in Table 4, which displays the results of the original paper (Qader et al., 2018), there is no significant correlation between redundancy and any metric at all. Considering that Qader et al. (2018) explicitly stated in the paper that the references were excluded when comparing the metric scores with the human judgements, we also computed the correlations once without the references. However, this attempt only led to a worse result (see Table 6), since none of the correlation values could be reproduced.

Since Qader et al. (2018) were not able to provide us with the original code or the corresponding information, it was impossible to determine the reason for the difference. For this reason and the consequent non-comparability of the results, we have refrained from reproducing the correlation matrix using the human evaluation results obtained in this replication study.

4 Conclusion

In this replication, we could not reproduce all results of the original study "Generation of Company descriptions using concept-to-text and text-to-text deep models: dataset collection and systems evalu-

	BLEU	NIST	METEOR	ROUGE-L	CIDEr	Coverage	Redundancy	Semantic
NIST	0.64							
METEOR	0.29	0.00						
ROUGE-L	0.11	-0.07	0.96					
CIDEr	0.00	-0.14	0.86	0.93				
Coverage	0.64	0.21	0.64	0.50	0.39			
Redundancy	-0.14	0.00	0.36	0.54	0.64	0.25		
Semantic	-0.14	-0.43	0.71	0.82	0.93	0.25	0.64	
Grammar	-0.38	-0.38	0.59	0.74	0.90	0.00	0.61	0.92

Table 4: ORIGINAL: Correlation matrix from Qader et al. (2018), human vs. automatic metric correlations are in the black square. Color markers indicate significant correlations, the different colors are for better readability

	BLEU	NIST	METEOR	ROUGE-L	CIDEr	Coverage	Redundancy	Semantic
NIST	0.64							
METEOR	0.29	0.00						
ROUGE-L	0.11	-0.07	0.96					
CIDEr	0.00	-0.14	0.86	0.93				
Coverage	0.57	0.28	0.29	0.21	0.18			
Redundancy	-0.04	-0.11	0.89	0.96	0.96	0.25		
Semantic	-0.21	-0.36	0.36	0.54	0.71	0.25	0.64	
Grammar	-0.45	-0.31	0.23	0.45	0.68	0.00	0.61	0.92

Table 5: REPLICATION: Correlation matrix reproduced based on the human evaluation results from Qader et al. (2018), computed at the system-level (**including** reference), using automatic metric scores from Table 7 in Appendix A.3. Color markers indicate significant correlations, the different colors are for better readability.

	BLEU	NIST	METEOR	ROUGE-L	CIDEr	Coverage	Redundancy	Semantic
NIST	0.43							
METEOR	-0.14	-0.60						
ROUGE-L	-0.43	-0.71	0.94					
CIDEr	-0.6	-0.83	0.77	0.89				
Coverage	0.31	-0.14	-0.14	-0.26	-0.31			
Redundancy	-0.66	-0.77	0.83	0.94	0.94	-0.20		
Semantic	-0.60	-0.82	0.09	0.31	0.60	0.09	0.49	
Grammar	-0.82	-0.60	-0.03	0.26	0.60	-0.25	0.49	0.89

Table 6: REPLICATION: Correlation matrix reproduced based on the human evaluation results from Qader et al. (2018), computed at the system-level (**excluding** reference), using automatic metric scores from Table 7 in Appendix A.3. Color markers indicate significant correlations, the different colors are for better readability.

ation" of Qader et al. (2018)

The primary goal of ReproGen (Belz et al., 2021) was to conduct an equivalent human evaluation with the aim of obtaining comparable values. We were able to reproduce the human evaluation and obtain results that are not only apparently comparable but also to obtain a moderate to strong statistical significant correlation (Taylor, 1990; Schober et al., 2018) using both Spearman's ρ and Pearson's r. However, this has taken a lot of time to gather all the information needed from both the paper and the authors.

In contrast to the first one, our secondary objective, namely to investigate whether we could obtain comparable inferences with the reproduced correlation matrix based on our human evaluation results, was not successful. We had to make several assumptions of missing information and even with that, we were not even able to recalculate the original results by using the human evaluation results from Qader et al. (2018). Therefore, we have refrained from a comparison with our data.

References

Anya Belz, Shubham Agarwal, Anastasia Shimorina, and Ehud Reiter. 2021. The reprogen shared task on reproducibility of human evaluations in nlg: Overview and results. In *Proceedings of the 14th International Conference on Natural Language Generation*. Website of the shared Task: https://reprogen.github.io/, last accessed: August 30, 2021.

Anya Belz, Simon Mille, and David M. Howcroft. 2020. Disentangling the properties of human evaluation methods: A classification system to support comparability, meta-evaluation and reproducibility testing. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 183–194, Dublin, Ireland. Association for Computational Linguistics.

K. Bretonnel Cohen, Jingbo Xia, Pierre Zweigenbaum, Tiffany Callahan, Orin Hargraves, Foster Goss, Nancy Ide, Aurélie Névéol, Cyril Grouin, and Lawrence E. Hunter. 2018. Three dimensions of reproducibility in natural language processing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Cédric Fayet, Alexis Blond, Grégoire Coulombel, Claude Simon, Damien Lolive, Gwénolé Lecorvé, Jonathan Chevelu, and Sébastien Le Maguer. 2020. FlexEval, création de sites web légers pour des campagnes de tests perceptifs multimédias. In *6e*

conférence conjointe Journées d'Études sur la Parole (JEP, 31e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition), pages 22–25, Nancy, France. ATALA.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from reproduction problems: What replication failure teaches us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1691–1701, Sofia, Bulgaria. Association for Computational Linguistics.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. 2020. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program).

Raheel Qader, Khoder Jneid, François Portet, and Cyril Labbé. 2018. Generation of company descriptions using concept-to-text and text-to-text deep models: dataset collection and systems evaluation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 254–263, Tilburg University, The Netherlands. Association for Computational Linguistics.

Edward Raff. 2019. A step toward quantifying independently reproducible machine learning research. *Advances in Neural Information Processing Systems*, 32:5485–5495.

Patrick Schober, Christa Boer, and Lothar A Schwarte. 2018. Correlation coefficients: appropriate use and interpretation. *Anesthesia & Analgesia*, 126(5):1763–1768.

Richard Taylor. 1990. Interpretation of the correlation coefficient: a basic review. *Journal of diagnostic medical sonography*, 6(1):35–39.

Martijn Wieling, Josine Rawee, and Gertjan van Noord. 2018. Squib: Reproducibility in computational linguistics: Are we willing to share? *Computational Linguistics*, 44(4):641–649.

A Appendix

A.1 Resources

This section lists the external resources that were used and describes whether they were made available in advance or have been collected during the implementation.

- The original paper (Qader et al., 2018), included by the task of ReproGen (Belz et al., 2021).
- The Human Evaluation Datasheet v1.0 [4] (Belz et al., 2020) filled out by Qader et al. (2018) (incomplete), included by the task of ReproGen (Belz et al., 2021).
- The web based evaluation tool "FlexEval" (Fayet et al., 2020), included by the task of ReproGen (Belz et al., 2021).
- Google Forms[5], used to do the survey.
- The part of the Wikipedia data sets that was used for the human evaluation, provided upon request by Qader et al. (2018).
- The anonymized original human evaluation results, provided upon request by Qader et al. (2018).

A.2 Access to Resources

The data we are able to publish, including code and results, is available in this Github Repository: https://github.com/der-Richter/TUDA-Reproducibility-ReproGen. To obtain access to the data from the original study, please contact Qader et al. (2018) directly.

A.3 Tables

System	BLEU	NIST	METEOR	ROUGE_L	CIDEr
C2T	0.0608	1.9322	0.0906	0.2092	0.1872
C2T_char	0.0750	1.0975	0.1159	0.2665	0.2731
C2T+pg	0.0413	0.0893	0.1076	0.2668	0.2836
C2T+pg+cv	0.0490	0.2349	0.1045	0.2589	0.2734
T2T+pg	0.0567	1.9690	0.1002	0.2212	0.1992
T2T+pg+cv	0.0558	2.1188	0.1024	0.2216	0.1974

Table 7: System results on the test set of the Wikipedia Company Corpus from Qader et al. (2018)

[4] https://drive.google.com/file/d/1_74CJ_n8vSPm8FvA6P_Sg49aZp3kecRo/view

[5] https://forms.gle/AQJS2s2GAHKAKPgd9

DialogSum Challenge: Summarizing Real-Life Scenario Dialogues

Yulong Chen[♠♡], **Yang Liu**[♣], **Yue Zhang**[♡♢]
[♠] Zhejiang University, China
[♡] School of Engineering, Westlake University, China
[♣] ILCC, School of Informatics, University of Edinburgh, UK
[♢] Institute of Advanced Technology, Westlake Institute for Advanced Study, China
yulongchen1010@gmail.com inf.yangl@outlook.com
yue.zhang@wias.org.cn

Abstract

We propose a shared task on summarizing real-life scenario dialogues, *DialogSum Challenge*, to encourage researchers to address challenges in dialogue summarization, which has been less studied by the summarization community. Real-life scenario dialogue summarization has a wide potential application prospect in chat-bot and personal assistant. It contains unique challenges such as special discourse structure, coreference, pragmatics and social common sense, which require specific representation learning technologies to deal with. We carefully annotate a large-scale dialogue summarization dataset based on multiple public dialogue corpus, opening the door to all kinds of summarization models.

Dialogue from DIALOGSUM:
#Person_1#: Good morning. I wonder whether you have got an answer from your superior.
#Person_2#: Yes, we had a meting about it yesterday afternoon.
#Person_1#: What's the answer?
#Person_2#: We decided that we could agree to your price, but we are a bit worried about the slow delivery.
#Person_1#: Let me see. I quoted your delivery in three months, didn't I?
#Person_2#: Yes, but we hope that the wool could reach us as soon as possible.
#Person_1#: I thought you would. So I rang Auckland last night. As you are our biggest customer, they agreed to ship the order on the first vessel available that will leave Auckland next month.
#Person_2#: Good, if you agree we'll draft the agreement right away and sign it then.
#Person_1#: By all means.

Summary from DIALOGSUM: #Person_1# and #Person_2# agree to sign an agreement ***since*** #Person_1# could speed up the delivery as #Person_2# hopes.

Figure 1: An example from DIALOGSUM dataset.

1 Task Overview

The *DialogSum Challenge* asks a model to generate a salient, concise, fluent, and coherent summary, given a piece of multi-turn dialogue text. The dialogue summary is highly abstractive in nature, and is supposed to be objective compared with monologue summarization. We will conduct both automatic and manual blind evaluation on the submitted models. In particular, to address unique challenges in dialogue summarization, we will manually evaluate system-generated summaries from multiple aspects designed for dialogue summarization, including coreference information, discourse relation, intent identification and objective description. An example is shown in Figure 1, where the summary describes main events in a business conversation.

2 Motivation

Thanks to the advance in neural network models, and availability of large scale labeled datasets, recent research has achieved promising progress on summarizing monologic texts, such as news articles (Liu and Lapata, 2019; Gehrmann et al., 2018), patents (Pilault et al., 2020) and academic papers (Koncel-Kedziorski et al., 2019). However, dialogue, as an important channel for achieving communicative intents, differs from monologic texts in nature and has received significantly less attention from the summarization research community. A major reason is the paucity of suitable dialogue summarization datasets.

To cope with this problem, we build a large scale labeled summarization dataset for real-life scenario dialogues, DIALOGSUM (Chen et al., 2021). An example from DIALOGSUM is shown in Figure 1. Compared with existing dialogue summariztaion datasets (Carletta et al., 2005; Gliwa et al., 2019; Zhong et al., 2021; Zhu et al., 2021), DIALOGSUM is useful for training neural models and is staying in the spoken domain as opposed to the written chat domain. Also, it contains diverse task-oriented dialogues that cover a wide range of daily-life topics. Summarizing those dialogues is useful for both business (e.g. help a business find common needs) and personal uses (e.g. track important events as

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 308–313,
Aberdeen, Scotland, UK, 20-24 September 2021.

Dialogue from DIALOGSUM:
#Person_1#: Good morning. What can I do for you?
#Person_2#: I'm in Room 309. ***I'm checking out today.*** Can I have my bill now?
#Person_1#: Certainly. Please wait a moment. Here you are.
#Person_2#: Thanks. ***Wait…***What's this? The 30 dollar for?
#Person_1#: ***Excuse me…*** The charge for your laundry service on Nov. 20th.
#Person_2#: But I didn't take any laundry service during my stay here. ***I think you have added someone else's.***
#Person_1#: ***Ummm…*** Sorry, would you mind waiting a moment? We check it with the department concerned.
#Person_2#: No. As long as we get this straightened out.
#Person_1#: I'm very sorry. There has been a mistake. We'll corrected the bill. Please take a look.
#Person_2#: ***Okay, here you are.***
#Person_1#: Goodbye.

Summary from DIALOGSUM: #Person_2# is checking out and asks #Person1# for the bill. #Person1# gives #Person_2# a ***wrong*** bill at first then corrects it.

Figure 2: Selected case from DIALOGSUM dataset.

Dialogue from DIALOGSUM:
#Person_1#: Hey, don't I know you from somewhere?
#Person_2#: No, sorry. I don't think so.
#Person_1#: Didn't you use to work at Common Fitness Gym?
#Person_2#: No, I'm afraid I did not.
#Person_1#: Oh, but I know you from somewhere else. Did you use to work at the movie theater downtown? You did. Yes. It's you. I go there all the time and you always sell me popcorn and soda.
#Person_2#: No, that's not me either. Sorry, ma'am. Perhaps I look familiar to you, but ...
#Person_1#: No, I know you. I have met you before! Hold on. Let me think. This is driving me crazy. I know that we've talked before. Oh, I remember now. You work at the Whole Bean Cafe on the corner. It that right?
#Person_2#: No, wrong again. Sorry, ma'am, but I really have to get going.

Summary from DIALOGSUM: #Person_1# **thinks that #Person_1# knows #Person_2# somewhere**, but #Person_2# denies it.

Figure 3: Selected case from DIALOGSUM dataset.

personal assistants). Empirical study and analysis demonstrate challenges in real-life scenario dialogue summarization (Chen et al., 2021).

To highlight the challenges in dialogue summarization, we propose real-life scenario dialogue summarization challenge, *DialogSum Challenge*, to encourage researchers to investigate such problems. The evaluation for dialogue summarization contains both automatic evaluation, i.e. ROUGE score (Lin, 2004) and BERTScore (Zhang et al., 2019), and human evaluation from multiple aspects to address corresponding challenges (c.f. Section 2.1 and Section 3.3.2). For human evaluation, we anonymize the submitted models, and evaluate them on corresponding hidden sub-test sets to ensure the fairness.

2.1 Unique Challenges in DIALOGSUM

Although dialogue summarization is in line with the philosophy of monologue summarization, we find some unique challenges in dialogue summarization.

First, because of special linguistic phenomena, the dialogue on the source side differs from monologue. Dialogue information flow is intuitively reflected in the **dialogue discourse structures** (Wolf and Gibson, 2005), where two utterances can be closely related even there is a large distance between them. Such a phenomenon is common in procedures and negotiations. For example, in Figure 1, the penultimate utterance "*... we'll draft the agreement... and sign it...*" is actually replying to the third utterance "*What's the answer?*", between which the utterances can be viewed as negotiation process and conditions. Also, frequent **coreference** and **ellipsis** make dialogue difficult to understand (Grosz et al., 1995; Quan et al., 2019). For example, to generate "wrong" in the summary in Figure 2, the model needs to understand "*I think you have added someone else's (laundry service on my bill)*", where "*my bill*" refers to "*#Person_2#'s bill*". These linguistic phenomena make dialogues difficult to encode using ordinary representation learning techonologies (Chen et al., 2021).

Second, compared with monologic summarization, dialogues are summarized from an *observer's* perspective, which requires summary to be **objective**. For example, in Figure 3, #Person_1#'s statements are actually awaiting to be confirmed. Human annotators identified such situation and used objective language ("*#Person_1# thinks that #Person_1#...*") to describe those statements. Also, the process of *perspective shift* (from interlocutor to observer) intuitively leads to morphology and lexical changes (e.g. the expression of referents and third-person singular predicates) and syntax changes (e.g. using written languages to describe spoken dialogues).

Third, dialogue summarization goes beyond summarizing dialogue contents, but also dialogue actions at the **pragmatic** level. For example, in the summary in Figure 1, "*agree*" summarizes both actions of #Person_1# and #Person_2#; in the summary in Figure 2, "*gives*" summarizes a single dialogue action of #Person_1#; in the summary in Figure 3, "*thinks*" and "*denies*" summarize multiple dialogue actions of #Person_1# and #Person_2#, respectively. It requires models

to not only summarize *what speakers are saying*, but also *what they are doing*.

3 Task Description

The task for participants is to provide a model that generates a summary given the input dialogue text. Both automatic and manual evaluation will be conducted to measure model performance.

3.1 Data

The participant of *DialogSum Challenge* can start immediately, as the DIALOGSUM dataset has been already public [1]. We collect 13,460 dialogue data for DIALOGSUM from three public dialogue corpora, namely Dailydialog (Li et al., 2017), DREAM (Sun et al., 2019) and MuTual (Cui et al., 2020), as well as an English speaking practice website. In term of size, DIALOGSUM is comparable with SAMSum while its average dialogue length is much longer than SAMSum, which comforts the purpose of summarization and is thus more challenging. The dialogue data cover a wide range of daily-life topics, including diverse task-oriented scenarios. We ask annotators to summarize the dialogue from an *observer's* perspective.

To ensure the annotation quality, each summary has been checked twice by different people, where the reward and punishment mechanism is included. We also sample and check the data after the second checking. When any error is found in the sampling checking, we ask annotators to repeat annotation and checking the annotation batch until no error can be found. To monitor the annotation and analyze inter-annotator agreement, we randomly select 500 dialogue, and ensure they are annotated and checked by different annotators. For each dialogue, we compare its summary and compute their pairwise ROUGE as shown in Table 1, which demonstrates our high annotation quality. Those 500 dialogues result in our test set. The public dataset consists of training (12,460), dev (500) and test (500) sets. For test set, we provide 3 references.

In addition to the public DIALOGSUM dataset, we build a ***hidden*** test set that consists of 100 dialogues and human annotated summaries. This ensures that participants will not be able to optimize their models against the hidden test set.

For the competition, participants can follow our data setting to train, develop and test their models on the public DIALOGSUM. Using external training data is allowed. For automatic evaluation, we will use both public and hidden test sets. For human evaluation, we will use the multiple subsets from Chen et al. (2021), which are collected from the test set, but not public.

Human Annotated Summary	R1	R2	RL
Summary1 to Summary2	52.90	26.01	50.42
Summary1 to Summary3	53.85	27.53	51.65
Summary2 to Summary3	53.30	26.61	50.44
Average	53.35	26.72	50.84

Table 1: ROUGE scores between three human annotated summaries in test set.

3.2 Protocol

Following previous work (Syed et al., 2018, 2020), we divide the competition into three phrases: (1) participants will train proposed summarization models using the provided training data on their hardware; (2) after submission system opens, participants will make their trained model submission to the TIRA. When the test data is available on the system, it will automatically make blind evaluation on the submitted model; (3) after the submission deadline is due, we will start to evaluate summaries generated by the final submitted models via crowdsourcing workers from multiple aspects.

We plan the following schedule for *DialogSum Challenge*. Please note that dates may be modified when we know the detailed schedule of INLG 2022.

- **20th September, 2021:** The shared task announced along with data available; call for participants.
- **20th Dec, 2021:** The submission system and public leaderboard open; participants can submit trained models to the TIRA infrastructure; the TIRA infrastructure will automatically evaluate submitted models with automatic metrics; the online leaderboard will keep updating the best performance on both public test set and hidden test set.
- **20th Feb, 2022:** The deadline for final model submissions; manual evaluation via crowdsourcing begins.

3.3 Evaluation

Our evaluation contains both automatic evaluation metric and human evaluation.

[1] https://github.com/cylnlp/DialogSum

3.3.1 Automatic Evaluation

We will report ROUGE and BERTScore (Zhang et al., 2019). ROUGE measures the overlap of n-grams in the generated summary against the reference summary, intuitively reflecting model's capturing ability of salient information. We will use ROUGE as the main automatic evaluation metric. BERTScore computes a similarity score between the generated summary and reference summary on token level using contextual embeddings, which provides a more robust evaluation method for generation tasks. We will use BERTScore as a supplementary metric. We will report the lowest, highest and averaged scores on our multi-reference test set, to better evaluate model performance, including their variance.

3.3.2 Human Evaluation

We previsouly show that, although models can achieve high ROUGE scores, their generated summaries can contain many errors regarding dialogue understanding. Thus, we design human evaluation from multiple aspect based on Chen et al. (2021). To ensure the fairness, we will conduct human evaluation via Amazon Mechanical Turk, and each generated summary will be judged by three annotators to ensure the accuracy. All human annatators will read system-generated summaries and rate them based on following criteria.

Standard Summarization Metrics: Fluency, Consistency, Relevance and Coherence Following Kryscinski et al. (2019, 2020), we evaluate system-generated summaries from four dimensions, which have been widely used as standard summary evaluation criteria in human evaluation for monologue text. Human annotators will follow Kryscinski et al. (2019)'s criteria, and evaluate on a 50 randomly selected sub-testset.

Coreference Information Chen et al. (2021) find that a big challenge in dialogue summarization is that, because of interactive information flow, models show poor performance on correctly aligning interlocutors and their conversation actions/contents. Thus, we will ask human annotators to follow Chen et al. (2021)'s criteria and rate the summary on a 50 randomly selected sub-testset.

Intent Identification A comprehensive dialogue summary expresses interlocutors' intents (i.e. the function of their utterances), which is frequent in dialogues and essential to understanding dialogues. However, system-generated summaries usually focus on the consequence of a dialogue, and fail to correctly identify interlocutors' intents. Therefore, we will conduct human evaluation on intent identification on the 50 randomly selected sub-testset following Chen et al. (2021).

Discourse Relation Coherent summaries convey important relations between main events, and identifying discourse relations and using proper phrases to express them can be challenging for summarization systems (Xu et al., 2020). However, causally related events are usually not explicitly expressed, and the distance between them is long due to the unique dialogue discourse structure (Grosz et al., 1995). To quantify such challenge, we will conduct human evaluation on discourse relation following (Chen et al., 2021) on the discourse sub-testset.

Objective Description In addition to the above evaluation aspects, we also find that models tend to take all interlocutors' contents as ground truth while failing to reason whether their statements are just subjective assumptions or even defended to be fake. Therefore, we will evaluate whether system-generated summaries use objective language to describe dialogues, and give scores from -1, 0, 1, where 1 means all correct, 0 means partially correct and -1 means all incorrect.

Overview Score To give an overview score for each model, we will ask annotators to evaluate each summary along with the above multi-aspect evaluation scores and give a score from 1 to 5. The higher, the better.

3.3.3 Overview Ranking

As mentioned, models that achieve high performance regarding automatic evaluation still contain many errors. Thus, the final ranking will be determined by human annotators' judgements. However, as our human evaluation contains multiple aspects and the cost can be high, we will only conduct human evaluation on a limited number of candidate models, which show leading performance on automatic evaluation metrics against the hidden test set. Up to the top twenty submission will be considered as candidate model for human evaluations.

4 Conclusion

Different from existing summarization datasets, the DIALOGSUM poses unique challenges in dialogue summarization. And we believe that *DialogSum*

Challenge will open new avenues for researchers to investigate solutions and study the linguistic phenomena in dialogue summarization.

5 Ethics Consideration

Dialogue data of *DialogSum Challenge* are collected from DailyDialog, DREAM, MuTual and an English practicing website that all are public to academic use and do not contain any personal sensitive information.

The construction of additional *DialogSum Challenge* hidden test set involves manual annotation. We ask annotators to write summarize limited to given dialogues, thus no personal or sensitive information is introduced.

References

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. 2005. The ami meeting corpus: A pre-announcement. In *International workshop on machine learning for multimodal interaction*, pages 28–39. Springer.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. DialogSum: A real-life scenario dialogue summarization dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5062–5074, Online. Association for Computational Linguistics.

Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang, and Ming Zhou. 2020. MuTual: A dataset for multi-turn dialogue reasoning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1406–1416, Online. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China. Association for Computational Linguistics.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China. Association for Computational Linguistics.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Chris Pal. 2020. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319, Online. Association for Computational Linguistics.

Jun Quan, Deyi Xiong, Bonnie Webber, and Changjian Hu. 2019. GECOR: An end-to-end generative ellipsis and co-reference resolution model for task-oriented dialogue. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4547–4557, Hong Kong, China. Association for Computational Linguistics.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A challenge data set and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7:217–231.

Shahbaz Syed, Wei-Fan Chen, Matthias Hagen, Benno Stein, Henning Wachsmuth, and Martin Potthast. 2020. Task proposal: Abstractive snippet generation for web pages. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 237–241.

Shahbaz Syed, Michael Völske, Martin Potthast, Nedim Lipka, Benno Stein, and Hinrich Schütze. 2018. Task proposal: The tl; dr challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 318–321.

Florian Wolf and Edward Gibson. 2005. Representing discourse coherence: A corpus-based study. *Computational Linguistics*, 31(2):249–287.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5021–5031.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938*.

Chenguang Zhu, Yang Liu, Jie Mei, and Michael Zeng. 2021. Mediasum: A large-scale media interview dataset for dialogue summarization. *arXiv preprint arXiv:2103.06410*.

Quality Evaluation of the Low-Resource Synthetically Generated Code-Mixed Hinglish Text

Vivek Srivastava
TCS Research
Pune, Maharashtra, India
srivastava.vivek2@tcs.com

Mayank Singh
IIT Gandhinagar
Gandhinagar, Gujarat, India
singh.mayank@iitgn.ac.in

Abstract

In this shared task, we seek the participating teams to investigate the factors influencing the quality of the code-mixed text generation systems. We synthetically generate code-mixed Hinglish sentences using two distinct approaches and employ human annotators to rate the generation quality. We propose two subtasks, *quality rating prediction* and *annotators' disagreement prediction* of the synthetic Hinglish dataset. The proposed subtasks will put forward the reasoning and explanation of the factors influencing the quality and human perception of the code-mixed text.

1 Introduction

Code-mixing is the phenomenon of mixing words and phrases from multiple languages in a single utterance of a text or speech. Figure 1 shows the example code-mixed Hinglish sentences generated from the corresponding parallel Hindi and English sentences. Code-mixed languages are prevalent amongst multilingual communities such as Spain, India, and China. With the inflation of social-media platforms in these communities, the availability of code-mixed data is seeking a boom. It has lead to several interesting research avenues for problems in computational linguistics such as language identification (Singh et al., 2018; Shekhar et al., 2020), machine translation (Dhar et al., 2018; Srivastava and Singh, 2020), language modeling (Pratapa et al., 2018), etc.

Over the years, we observe various computational linguistic conferences and workshops organizing the shared tasks involving the code-mixed languages. Diverse set of problems have been hosted such as sentiment analysis (Chakravarthi et al., 2021; Patwa et al., 2020), offensive language identification (Chakravarthi et al., 2021), word-level language identification (Solorio et al., 2014; Molina et al., 2016), information retrieval (Banerjee et al., 2016), etc.

Example I

HINGLISH: ye ek code mixed sentence ka example hai
ENGLISH : this is an example code-mixed sentence

Example II

HINGLISH : kal me movie dekhne ja raha hu. How are the reviews?
ENGLISH: I am going to watch the movie tomorrow. How are the reviews?

Figure 1: Example parallel Hinglish and English sentences. The code-mixed Hinglish sentences contain words from Hindi and English languages.

Despite these overwhelming attempts, the natural language generation (NLG) and evaluation of the code-mixed data remain understudied. The noisy and informal nature of the code-mixed text adds to the complexity of solving and evaluating the various NLG tasks such as summarization and machine translation. These inherent challenges (Srivastava and Singh, 2020) with the code-mixed data makes the widely popular evaluation metrics like BLEU and WER obsolete. Various metrics (e.g., CMI (Das and Gambäck, 2014; Gambäck and Das, 2016), M-index (Barnett et al., 2000), I-index (Guzmán et al., 2017), Burstiness (Goh and Barabási, 2008), Memory (Goh and Barabási, 2008), etc.) have been proposed to measure the complexity of code-mixed data, but they fail to capture the linguistic diversity which leads to poorly estimating the quality of code-mixed text (Srivastava and Singh, 2021a).

With this shared task[1] (see Section 2 and 4 for the detailed description), we look forward to the

[1] https://sites.google.com/view/hinglisheval

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 314–319,
Aberdeen, Scotland, UK, 20-24 September 2021.

new strategies that cater to the broad requirement of the quality evaluation of the generated code-mixed text. These methods will entail various linguistic features encompassing syntax and semantics and the perspectives of human cognition such as writing style, emotion, sentiment, language, and preference. We also put forward a subtask to understand the factors influencing the human disagreement on the quality rating of the generated code-mixed text. This could help design a more robust quality evaluation system for the code-mixed data.

2 Task Overview

In this shared task, we propose two subtasks evaluating the quality of the code-mixed Hinglish text. First, we propose to predict the quality of Hinglish text on a scale of 1–10. We aim to identify the factors influencing the text's quality, which will help build high-quality code-mixed text generation systems. We synthetically generate the Hinglish sentences using two different approaches (see Section 3) leveraging popular English-Hindi parallel corpus. Besides, we also have at least two human-generated Hinglish sentences corresponding to each parallel sentence. The second subtask aims to predict the disagreement on a scale of 0–9 between the two annotators who have annotated the synthetically generated Hinglish sentences. Various factors influence this human disagreement, and we seek to investigate the reasoning behind this behavior.

3 Dataset

As outlined in Section 1, the code-mixed NLG task observes a scarcity of high-quality datasets. Consequently, the quality evaluation of the generated code-mixed text remains unexplored. We propose a new dataset with Hinglish sentences generated synthetically and rated by human annotators to address this challenge. We create the dataset in two phases.

Phase 1: Human-generated Hinglish sentences: We select the English-Hindi parallel sentences from the IIT-B parallel corpus(Kunchukuttan et al., 2018) to generate the Hinglish sentences. The parallel corpus has 1,561,840 sentence pairs. We randomly select 5,000 sentence pairs, in which the number of tokens in both the sentences is more than five. We employ five human annotators and assign each 1,000 sentence pairs. Table 1 shows the annotation guidelines to generate the Hinglish sentences. Post annotation, we obtain 1,976 sentence pairs for which the annotators have generated at least two Hinglish sentences.

Phase 2: Synthetic Hinglish sentence generation and quality evaluation: We synthetically generate the Hinglish sentence corresponding to each of the parallel 1,976 English-Hindi sentence pairs. We employ two different code-mixed text generation (CMTG) techniques:

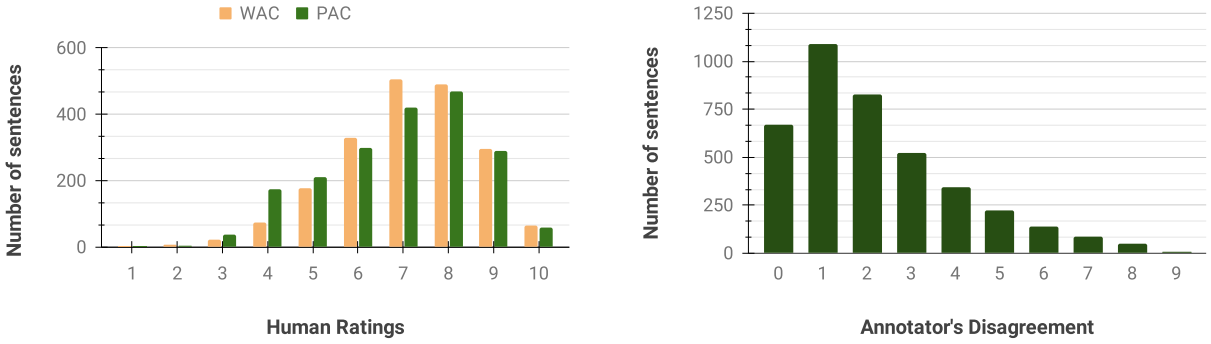
- Word-aligned CMTG (WAC): Here, we align the noun and adjective tokens between the parallel sentences. We replace the aligned Hindi token with the corresponding English token and transliterate the Hindi sentence to the Roman script.
- Phrase-aligned CMTG (PAC): Here, we align the key-phrases of length up to three tokens between the parallel sentences. We replace the aligned Hindi phrase with the corresponding English phrase and transliterate the Hindi sentence to the Roman script.

For the token alignment between parallel sentences, we use the online curated dictionaries, GIZA++ (Och and Ney, 2003) trained on the remaining IIT-B corpus, and cross-lingual word embedding trained on English and Hindi word vectors from FastText (Bojanowski et al., 2017). We employ eight human annotators[2] to provide a rating between 1 (low quality) to 10 (high quality) to the generated Hinglish sentences. Table 1 shows the annotation guidelines to rate the sentences. Figure 2a and 2b shows the distribution of the annotators' rating and their disagreement, respectively.

Data format: Table 2 shows an instance from the dataset. In total, we have 3,952 instances[3] in the dataset where each data instance i for subtask-1 (see Section 4.1) is represented as $\mathbf{X1}_i$={Eng$_i$, Hin$_i$, Synthetic_Hing$_i$} and $\mathbf{y1}_i$={Average_rating$_i$}. For subtask-2 (see Section 4.2), the instance j is represented as $\mathbf{X2}_j$={Eng$_j$, Hin$_j$, Synthetic_Hing$_j$} and $\mathbf{y2}_j$={Annotator_disagreement$_j$}. In addition, we provide at least two human generated Hinglish sentences corresponding to each data instance for both the subtasks. We shuffle and split the dataset in the ratio 70:10:20 with 2766, 395, and 791 data instances in train, validation, and test respectively. The more detailed description of the dataset is available in (Srivastava and Singh, 2021b).

[2]Different from the annotators in Phase 1. Each annotator gets 247 sentences generated by PAC and WAC, each corresponding to the same set of parallel sentences.

[3]Two synthetic Hinglish sentences are generated for each parallel sentence pair.



(a) (b)

Figure 2: Distribution of (a) human evaluation scores and (b) disagreement in human scores in the synthetically generated Hinglish sentences.

Task	Guidelines
Hinglish text generation	1. The Hinglish sentence should be written in Roman script. 2. The Hinglish sentence should have words from both the source languages. 3. Avoid using new words, wherever possible, that are not present in both sentences. 4. If the source sentences are not the translation of each other, mark the sentence pair as "#".
Quality rating	The rating depends on the following three factors: 1. The similarity between the generated Hinglish sentence and the source sentences. 2. The readability of the generated sentence. 3. The grammatical correctness of the generated sentence.

Table 1: Annotation guidelines to the annotators for the two different tasks.

4 The Two Tasks

4.1 Subtask 1: Quality rating prediction

The first subtask is predicting the quality rating of the code-mixed text. The participating teams can use the English, Hindi, and human-generated Hinglish sentences to predict the average rating[4] as provided by the human annotators to the synthetic Hinglish sentences. In addition, we seek the teams to answer the following research questions implicitly with their experiments (not an exhaustive list):

- **RQ1.1**: Do the quality of source English and Hindi sentences impact Hinglish sentences' quality?
- **RQ1.2**: Does the quality of Hinglish text generated by humans has any correlation with the quality of Hinglish text generated synthetically?
- **RQ1.3**: Does the dominance of a language (English or Hindi) present in the Hinglish sentence impact the rating provided by the humans?
- **RQ1.4**: How does the semantic and the syntactic correctness of the Hinglish sentence influence its quality?

[4]We take the greatest integer i $\leq$ average of the two rating scores.

4.2 Subtask 2: Annotators' disagreement prediction

The next subtask is predicting the disagreement between the ratings provided by the human annotators to the synthetic Hinglish sentences. We calculate the disagreement between the ratings as the absolute difference between the two rating scores. Additionally, we seek the participating teams to answer the following research questions implicitly with their experiments (not an exhaustive list):

- **RQ2.1**: Does the quality of sentences in the source languages (English and Hindi) have any influence on the quality of the synthetic Hinglish sentences as seen by different individuals?
- **RQ2.2**: Does the quality of human-generated Hinglish sentence has any correlation with the quality of synthetic Hinglish text as seen by different individuals?
- **RQ2.3**: Do humans have a language bias while rating the quality of the code-mixed text?
- **RQ2.4**: Do the similarity between human-generated and synthetic Hinglish sentences influence the annotators' disagreement?

English	Hindi	Human-generated Hinglish	Synthetic Hinglish 1	Synthetic Hinglish 2
The reward of goodness shall be nothing but goodness.	अच्छाई का बदला अच्छाई के सिवा और क्या हो सकता है?	The reward of achai shall be nothing but achai. Goodness ka badla goodness ke siva aur kya ho sakta hai. Achai ka badla shall be nothing but achai.	reward ka badla reward ke nothing aur kya ho sakta hai **Rating1**: 7 **Rating2**: 4	reward of goodness goodness ke siva aur kya ho sakta hai **Rating1**: 9 **Rating2**: 7

Table 2: Example human-generated and synthetic Hinglish sentences from the dataset along with the source English and Hindi sentences. Two different human annotators rate the synthetic Hinglish sentences on the scale 1-10 (low-high quality).

5 Evaluation

We use the following three evaluation metrics:

- **F1-score (FS)**: We use the weighted F1-score to evaluate the system performance. The score ranges from 0 (worst) to 1 (best).
- **Cohen's Kappa (CK)**: We use the Cohen's Kappa score to measure the agreement between the predicted and the actual rating. The score ranges from ≤ 0 (high disagreement) to 1 (high agreement).
- **Mean Squared Error (MSE)**: MSE suggests the difference between the actual and the predicted scores. A low MSE score is preferred, with zero being the lowest possible score.

For the first subtask, we use all three metrics, whereas we use FS and MSE to evaluate the second subtask.

6 Pilot Experiment

We conducted a simple pilot experiment with a SOTA multilingual contextual language model M-BERT (Devlin et al., 2019). We fine-tune the pre-trained M-BERT model by adding one hidden-layer neural network on the top. We use the Relu activation function, AdamW optimizer with 0.03 learning rate, cross-entropy loss, and a batch size of 32. We use the contextual word-embedding corresponding to the synthetic Hinglish sentences in the dataset as an input to the model. The architecture remains the same for both subtasks.

Table 3 shows the result of the baseline experiment. We observe that the fine-tuned version of M-BERT performs poorly on both the subtasks on all the evaluation metrics. These language models are not as effective for both the subtasks as compared to other code-mixed text classification tasks where they seem to perform better than other rule-based and neural approaches (Gupta et al., 2021; Winata et al., 2021). Overall, we observe the poor performance of M-BERT based classifier on the current two subtasks. Specifically, for subtask 1, the agreement (measured by CK score) between predicted rating and human rating is close to 0. These results present an excellent opportunity to propose a shared task that enhances the generated code-mixed text quality estimation.

	Subtask 1			Subtask 2	
	FS	**CK**	**MSE**	**FS**	**MSE**
Val	0.202	0.003	2.797	0.209	4.987
Test	0.256	0.092	2.628	0.242	4.317

Table 3: Evaluation of the pilot experiment.

7 Conclusion

In contrast to the non-code-mixed text, the noisy and informal nature (e.g., spelling variation, missing punctuation, and language switching) of the code-mixed text makes the quality evaluation more loosely defined. Consequently, we need to build models that can effectively gauge the human perception of the quality of the code-mixed text. This shared task will help to build efficient and robust code-mixed text generation and evaluation systems.

References

Somnath Banerjee, Kunal Chakma, Sudip Kumar Naskar, Amitava Das, Paolo Rosso, Sivaji Bandyopadhyay, and Monojit Choudhury. 2016. Overview of the mixed script information retrieval (msir) at fire-2016. In *Forum for Information Retrieval Evaluation*, pages 39–49. Springer.

Ruthanna Barnett, Eva Codó, Eva Eppler, Montse Forcadell, Penelope Gardner-Chloros, Roeland van Hout, Melissa Moyer, Maria Carme Torras, Maria Teresa Turell, Mark Sebba, Marianne Starren, and Sietse Wensing. 2000. The lides coding manual: A document for preparing and analyzing language interaction data version 1.1—july, 1999. *International Journal of Bilingualism*, 4(2):131–132.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with

subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bharathi Raja Chakravarthi, Ruba Priyadharshini, Navya Jose, Anand Kumar M, Thomas Mandl, Prasanna Kumar Kumaresan, Rahul Ponnusamy, Hariharan R L, John P. McCrae, and Elizabeth Sherly. 2021. Findings of the shared task on offensive language identification in Tamil, Malayalam, and Kannada. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 133–145, Kyiv. Association for Computational Linguistics.

Amitava Das and Björn Gambäck. 2014. Identifying languages at the word level in code-mixed indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 378–387.

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

Mrinal Dhar, Vaibhav Kumar, and Manish Shrivastava. 2018. Enabling code-mixed translation: Parallel corpus creation and mt augmentation approach. In *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*, pages 131–140.

Björn Gambäck and Amitava Das. 2016. Comparing the level of code-switching in corpora. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1850–1855.

K-I Goh and A-L Barabási. 2008. Burstiness and memory in complex systems. *EPL (Europhysics Letters)*, 81(4):48002.

Akshat Gupta, Sai Krishna Rallabandi, and Alan W Black. 2021. Task-specific pre-training and cross lingual transfer for sentiment analysis in dravidian code-switched languages. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 73–79.

Gualberto A Guzmán, Joseph Ricard, Jacqueline Serigos, Barbara E Bullock, and Almeida Jacqueline Toribio. 2017. Metrics for modeling code-switching across corpora. In *INTERSPEECH*, pages 67–71.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The iit bombay english-hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016. Overview for the second shared task on language identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 40–49.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, PYKL Srinivas, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 774–790.

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1543–1553.

Shashi Shekhar, Dilip Kumar Sharma, and MM Sufyan Beg. 2020. Language identification framework in code-mixed social media text based on quantum lstm—the word belongs to which language? *Modern Physics Letters B*, 34(06):2050086.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018. Language identification and named entity recognition in hinglish code mixed tweets. In *Proceedings of ACL 2018, Student Research Workshop*, pages 52–58.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, et al. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 62–72.

Vivek Srivastava and Mayank Singh. 2020. Phinc: A parallel hinglish social media code-mixed corpus for machine translation. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 41–49.

Vivek Srivastava and Mayank Singh. 2021a. Challenges and limitations with the metrics measuring the complexity of code-mixed text. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 6–14.

Vivek Srivastava and Mayank Singh. 2021b. Hinge: A dataset for generation and evaluation of code-mixed hinglish text. *arXiv preprint arXiv:2107.03760*.

Genta Indra Winata, Samuel Cahyawijaya, Zihan Liu, Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2021. Are multilingual models effective in code-switching? In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 142–153.

Shared Task on Feedback Comment Generation for Language Learners

Ryo Nagata
Konan University, Japan
JST, PRESTO, Japan
nagata-genchal@ml.hyogo-u.ac.jp.

Masato Hagiwara
Octanove Labs, USA
masato@octanove.com

Kazuaki Hanawa
RIKEN, Japan
Tohoku University, Japan
kazuaki.hanawa@riken.jp

Masato Mita
RIKEN, Japan
Tohoku University, Japan
masato.mita@riken.jp

Artem Chernodub
Grammarly
artem.chernodub@grammarly.com

Olena Nahorna
Grammarly
olena.nahorna@grammarly.com

Abstract

In this paper, we propose a generation challenge called *Feedback comment generation for language learners*. It is a task where given a text and a span, a system generates, for the span, an explanatory note that helps the writer (language learner) improve their writing skills. The motivations for this challenge are: (i) practically, it will be beneficial for both language learners and teachers if a computer-assisted language learning system can provide feedback comments just as human teachers do; (ii) theoretically, feedback comment generation for language learners has a mixed aspect of other generation tasks together with its unique features and it will be interesting to explore what kind of generation method is effective against what kind of writing rule. To this end, we have created a dataset and developed baseline systems to estimate baseline performance. With these preparations, we propose a generation challenge of feedback comment generation.

1 Introduction

Feedback comment generation for language learners is a task where given a text and a span, a system generates, for the span, an explanatory note that helps the writer (language learner) improve their writing skills (for convenience of explanation, the task will be abbreviated as *feedback comment generation*, hereafter). The target language(s) can be any language, but we limit ourselves to English input texts and English feedback comments in this challenge. As shown in Figure 1, a feedback comment is typically made about erroneous, unnatural, or problematic words in a given text so that the writer can understand why the present form is not good together with the underlying rule.

In this regard, feedback comment generation is related to grammatical error detection and correction. In many cases, however, it is not enough to just point out an error with its correct form in order to help language learners with writing learning. Instead, it is often essential for them to explain underlying rules, which makes the task different from the conventional grammatical error detection/correction. In other words, it is essential in feedback comment generation to include more information than grammatical error detection/correction provide.

At the same time, unconstrained generation would make the task infeasible in terms of system development and evaluation. With this in mind, we set some constrains to the task to be explored in this generation challenge as described in Section 2. For example, the input is limited to a sentence (and a span) instead of a text.

The motivations for this challenge are as follows. A practical motivation is already mentioned above. It will be useful if a computer-assisted language learning system can provide feedback comments just as human teachers do. Theoretically, feedback comment generation has a mixed aspect of other generation tasks together with its unique features as described in Section 3. It will be interesting to explore what kind of technique is effective against what kind of writing rule.

One of the goals of this challenge is to reveal how well we can generate feedback comments with existing techniques. There is a wide variety of choices as generation methods that are applicable

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 320–324,
Aberdeen, Scotland, UK, 20-24 September 2021.

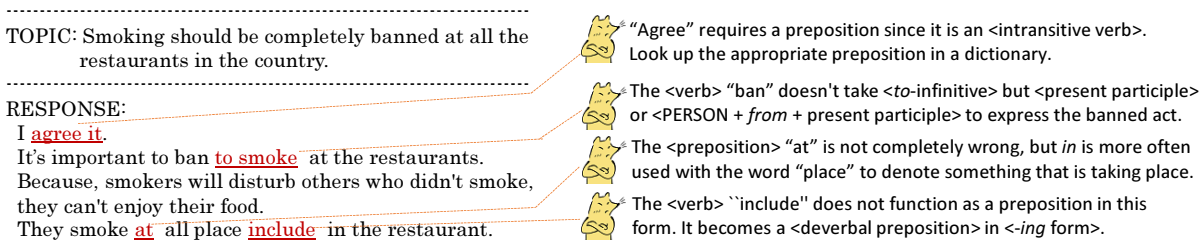


Figure 1: Example of Feedback Comments.

to this task. Nevertheless, they have not yet been explored (at least, much less than in other generation tasks). The generation challenge will enable the NLG community to evaluate and compare a range of techniques using the same dataset. Besides it will provide us with opportunities to develop new techniques.

Another goal is to shed a light on the difficulties in this task. This is going to be the first generation challenge of feedback comment generation as far as we are aware of. No one fully understands what is possible and impossible in the task. Holding this generation challenge will bring more insights into the task, which will in turn give new knowledge and experience to the NLG community.

Having said that, to make the task feasible within GenChal2021, we have prepared a dataset, evaluation metrics, and other necessities such as tools for this challenge as shown in Section 4. We have even developed baseline systems to estimate baseline performance. With these preparations, we propose a generation challenge of feedback comment generation.

2 Task Definition

2.1 General Definition

A unit of the input in feedback comment generation in general consists of a text and spans of the text. Spans, which are counted by 1-based index based on characters, correspond to where to comment. An example input text would be:

(1) *I agree it.*

as shown on the left-hand side of Figure 1. A span would be 3 to 10, which will be abbreviated as 3:10, hereafter.

The output for a span is a string that explains why the span is not good together with the underlying rule. To make the task different from grammatical error detection/correction, the output string has to contain more information than grammatical error detection/correction provide. In other words, just indicating the error position, the erroneous word(s), and/or the correct form are not enough as a valid feedback comment, of which details are discussed in Subsection 2.2.

2.2 Task Definition to Be Used

The above task definition is too general and abstract to be a practical one. For this reason, we put some constraints on it.

First, we limit the target only to preposition use as in the examples in Figure 1. It should be emphasized that the target includes missing prepositions, *to*-infinitives, and deverbal prepositions (e.g., *including*) in preposition use.

Second, we also limit the input to a narrower unit. Specifically, the input text always consists of only one sentence with one span. Also, they are pre-tokenized where tokens are separated by whitespace. For example, the first sentence in Figure 1 would give an input:

(2) *I agree it .* \t 3:10

where \t stands for the tab character. If a sentence contains more than one preposition error, it appears two or more times with different spans.

Under these settings, participants develop a system that automatically generates an appropriate feedback comment in English for an input sentence and a span. The length of a generated feedback comment should be less than 100 tokens. If a system cannot generate an appropriate feedback comment for a given span, it may generate the special token <NO_COMMENT>, which is not counted as a system output. This allows us to calculate recall, precision, and F_1 as explained below. An example output would be:

(3) *I agree it .* \t 3:10 \t **"agree" is an intransitive verb and thus it requires a preposition before its object.**

Also note that the input sentence and its span are included in the system output for evaluation convenience.

Evaluation is probably the hardest challenge in this task. We adopt automated and manual evaluation methods. In the former, we simply take BLEU between a system output and its corresponding reference (manually created feedback comment). In the latter, human evaluators examine whether a system output and its corresponding reference are equivalent in meaning. To be precise, a system output is regarded as appropriate if (1) it contains information similar to the reference and (2) it does not contain information that is irrelevant to the span; it may contain information that the reference does not contain as long as it is relevant to the span. This way of manual evaluation inevitably brings subjectivity to some extent. In practice, however, the results of a pilot study show that inter-evaluator agreement is considerably high as shown in Section 4.

The final manual evaluation measures are recall, precision, and F_1. Recall is defined as the number of appropriate system outputs divided by the number of target spans. Similarly, precision is defined as the number of appropriate system outputs divided by the number of system outputs where the special output <NO_COMMENT> is excluded. F_1 is the harmonic mean of recall and precision.

3 Related Work

Generally speaking, feedback comment generation is a task of text-to-text generation. It then can be abstractly regarded as a Machine Translation (MT) problem where the input text, which is written by a language learner, is translated into another text explaining writing rules. This implies that generation methods employed in MT or other related research areas may be effective in the present task. For example, feedback comments often refer to words and phrases appearing in the input text, and techniques for referring to words in the source text (e.g., copy mechanisms) will likely be beneficial.

Unlike MT, the equivalence between the source and target texts in meaning do not hold. Instead, the target text is a feedback comment that explains why the source is not good together with the underlying rule. From this point of view, the present task is related to research in dialogue systems.

Feedback comment generation has its unique aspects as well. It should be emphasized that a feedback comment is generated against a span (of the input text or sentence) whereas only a text (e.g., a sentence or utterance) is dealt with in other major text-to-text generation tasks such as MT and dialog systems. In consequence, feedback comment generation systems have to output different texts for the exact same source sentence, depending on given spans.

The source and target languages are also unique. In this challenge, both are English, but there is room for discussion whether they fall into the same language class. The former is learner English, and inevitably it contains erroneous/unnatural words. Even within correct sentences, grammar, expressions, and style are expected to be used differently from canonical English. This brings out further research questions related to the source and target languages. For example, which is the best setting of vocabularies — only one common vocabulary for the source and target, or one for each? Does a pre-trained general (or native) language model work well to model learner English? There are a number of unaddressed research questions like these.

Feedback comment generation is also related to grammatical error detection/correction. The state-of-the-art methods typically solve the problems as sequence labeling (e.g., Kaneko et al. (2017)) or MT with DNNs (e.g., Junczys-Dowmunt et al. (2018); Napoles and Callison-Burch (2017); Rothe et al. (2021)). Recently, a DNN-based sequence labeling method is combined with symbolic transformations (Omelianchuk et al., 2020), which can be a good source of information to generate feedback comments.

Some researchers (Kakegawa et al., 2000; McCoy et al., 1996; Nagata et al., 2014) made an attempt to develop rule-based methods for diagnosing errors in line with grammatical error correction. Researchers started to apply more modern techniques. Nagata (2019) showed that a neural-retrieval-based method was effective in preposition feedback comment generation. Lai and Chang (2019) proposed a method that used grammatical error correction and templates to generate detailed comments. Gkatzia et al. (2013) and Gkatzia et al. (2014) proposed methods for automatically choosing feedback templates based on learning history.

The availability of datasets for research in feedback comment generation has been increasing. Nagata (2019) released a dataset consisting of feed-

back comments on preposition use. They marked up erroneous prepositions and annotated them with feedback comments. Nagata et al. (2020) extended it to other grammatical errors and also other writing items such as discourse and lexical choice. Pilan et al. (2020) released a unique dataset where feedback comments on linking words were annotated.

4 Preparation

Based on the work (Nagata, 2019; Nagata et al., 2020), we created a new dataset for this generation challenge. The original texts are excerpts from the essays (written by learners of English) in ICNALE (Ishikawa, 2011). We had native speakers of English, who had experience in English teaching, manually annotated all preposition errors with feedback comments in English. Table 1 shows its statistics.

The dataset will be split into training, development, and test sets. Note that training and development sets consist of the whole essays, meaning that they contain all sentences no matter whether they contain feedback comments or not (i.e., error free essays are included in the sets). Also note that a sentence can be annotated with more than one feedback comment. In contrast, the test set only contains sentences with exactly one feedback comment each as described in Subsection 2.2. If a sentence contains more than one preposition error, it appears two or more times with different spans (in different lines). They will be provided for the participants in due course.

We also developed a Web-based submission system that takes system outputs the participants submit. The system checks the output format of the submission and calculate its score (i.e., BLEU).

We also implemented two baseline systems for this challenge. One is a deep neural network (DNN)-based retrieval system that retrieves the most similar instance to the input sentence and outputs it as a generation result. The other is a text generation system based on a DNN encoder-decoder with a copy mechanism.

As a pilot study, we tested them on the dataset (Nagata, 2019). For manual evaluation, we trained a professional annotator who had more than ten years of experience in English syntactic annotation and two years of experience in professional English writing teaching. The person and the first and third authors independently evaluated the generation results. They labeled each generated feedback comment as either *appropriate* or not, following the manner described in Subsection 2.2.

Corpus	ICNALE
Number of essays	1,884
Number of sentences	27,995
Number of tokens	473,815
Number of comments	5,237

Table 1: Statistics on Dataset.

As a result, it turned out that the retrieval system and the text generation system achieved an F_1 of 0.35 and 0.43, respectively[1]. Inter-evaluator agreements (Cohen's kappa coefficient) were considerably high, ranging from 0.86 to 0.92. These results imply that the present task is not easy one, but also not completely insolvable.

5 Organizers

The organizing group comprises six people as shown in the authors of this paper. All members have engaged in natural language research related to language learning and education for many years and some of them have organized several workshops and shared tasks in the domain.

The first author developed the dataset. The second author developed the submission system together with a Web page for this challenge. The two mainly designed this generation challenge with help from the other members. The third author implemented the baseline systems with the first author. They were also involved in the pilot manual evaluation.

All will be involved in the actual generation challenge including evaluation and paper publication. Although the trained professional evaluator is not included in the organizing group, the person will play a major role in manual evaluation.

6 Conclusions

In this paper, we have described a new generation challenge called *Feedback comment generation for language learners*. We have explored the task, describing the task definition, the related work, and the dataset to be used.

[1]The baseline systems are not designed to generate the special token <NO_COMMENT>, and they always output a feedback comment for a given span. Accordingly, it always holds that recall = precision =F_1.

References

Dimitra Gkatzia, Helen Hastie, Srinivasan Janarthanam, and Oliver Lemon. 2013. Generating student feedback from time-series data using reinforcement learning. In *Proc. of 14th European Workshop on Natural Language Generation*, pages 115–124.

Dimitra Gkatzia, Helen Hastie, and Oliver Lemon. 2014. Comparing multi-label classification with reinforcement learning for summarisation of time-series data. In *Proc. of 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1231–1240.

Shinichiro Ishikawa. 2011. *A new horizon in learner corpus studies: The aim of the ICNALE project*, pages 3–11. University of Strathclyde Publishing, Glasgow.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proc. of 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606.

Jun'ichi Kakegawa, Hisayuki Kanda, Eitaro Fujioka, Makoto Itami, and Kohji Itoh. 2000. Diagnostic processing of Japanese for computer-assisted second language learning. In *Proc. of 38th Annual Meeting of the Association for Computational Linguistics*, pages 537–546.

Masahiro Kaneko, Yuya Sakaizawa, and Mamoru Komachi. 2017. Grammatical error detection using error- and grammaticality-specific word embeddings. In *Proc. of 8th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 40–48.

Yi-Huei Lai and Jason Chang. 2019. TellMeWhy: Learning to explain corrective feedback for second language learners. In *Proc. of 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 235–240.

Kathleen F. McCoy, Christopher A. Pennington, and Linda Z. Suri. 1996. English error correction: A syntactic user model based on principled "mal-rule" scoring. In *Proc. of 5th International Conference on User Modeling*, pages 69–66.

Ryo Nagata. 2019. Toward a task of feedback comment generation for writing learning. In *Proc. of 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3197–3206.

Ryo Nagata, Kentaro Inui, and Shin'ichiro Ishikawa. 2020. Creating Corpora for Research in Feedback Comment Generation. In *Proc. of the 12th Language Resources and Evaluation Conference*, pages 340–345.

Ryo Nagata, Mikko Vilenius, and Edward Whittaker. 2014. Correcting preposition errors in learner English using error case frames and feedback messages. In *Proc. of 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 754–764.

Courtney Napoles and Chris Callison-Burch. 2017. Systematically adapting machine translation for grammatical error correction. In *Proc. of 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 345–356.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proc. of Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170.

Ildiko Pilan, John Lee, Chak Yan Yeung, and Jonathan Webster. 2020. A Dataset for Investigating the Impact of Feedback on Student Revision Outcome. In *Proc. of 12th Language Resources and Evaluation Conference*, pages 332–339.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A Simple Recipe for Multilingual Grammatical Error Correction. In *Proc. of 59th Annual Meeting of the Association for Computational Linguistics and 11th International Joint Conference on Natural Language Processing*, pages 702–707.

The SelectGen Challenge: Finding the Best Training Samples for Few-Shot Neural Text Generation

Ernie Chang[*], **Xiaoyu Shen**[*], **Alex Marin**[⋒], **Vera Demberg**
Dept. of Language Science and Technology, Saarland University
[⋒] Microsoft Corporation, Redmond, WA
{cychang,xshen}@coli.uni-saarland.de

Abstract

We propose a shared task on training instance selection for few-shot neural text generation. Large-scale pretrained language models have led to dramatic improvements in few-shot text generation. Nonetheless, almost all previous work simply applies random sampling to select the few-shot training instances. Little to no attention has been paid to the selection strategies and how they would affect model performance. The study of the selection strategy can help us to (1) make the most use of our annotation budget in downstream tasks and (2) better benchmark few-shot text generative models. We welcome submissions that present their selection strategies and the effects on the generation quality.

1 Introduction

Few-shot text generation is an important research topic since obtaining large-scale training data for each individual downstream task is prohibitively expensive. Recently, pretraining large neural networks with a language modeling objective has led to significant improvements across different few-shot text generation tasks (Radford et al., 2019; Lewis et al., 2020) and many techniques are proposed based on them (Chen et al., 2020; Schick and Schütze, 2020a; Zhang et al., 2020; Kale, 2020; Chang et al., 2020, 2021a,b; Li and Liang, 2021). However, all previous works simulate the few-shot scenario by randomly sampling a subset from the full training data. Little to no attention has been paid to the selection strategies.

The goal of the proposal is to call for innovative ideas on searching for an optimal strategy to select the few-shot training instances, as well as a comprehensive analysis of how the selection strategy would affect the model performance. The study of selection strategies is motivated by two rationales:

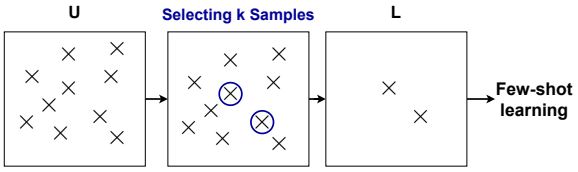


Figure 1: **Training scenario for few-shot text generation**: **U** represents unlabeled data and **L** indicates labeled instances. The annotation budget only allows selecting K data for annotating the reference text. We aim to identify the K most representative instances that, when annotated and trained on them, leads to a best model performance.

First, random sampling leads to a large variance of model performance (Zhang et al., 2020; Schick and Schütze, 2020a,b). Yet current works sample their own training data which makes it difficult to compare across different models. One can then not be sure whether an improved performance can be really ascribed to the model or the randomness of sampling. Using a stable selection strategy to find the most informative few-shot instances can provide a fair platform and better benchmark different few-shot generative models. Second, in practical applications, e.g. document summarization, the training data is usually obtained by manually annotating the summaries for some selected documents. In Figure 1, we illustrate the typical training scenario for text generation where the annotation budget only allows annotating a limited amount of data. Studying the optimal selection strategy can help make the most use of our annotation budget. Specifically, we focus on the label-free setting where *the selection can only condition on the unannotated data*. Although leveraging the reference text may benefit the selection strategy, it conflicts with the realistic setting where we need to first select the data then get its annotated reference text.

The selection task resembles the theme of active learning (Balcan et al., 2007; Chang et al., 2020, 2021c,a), where the model keeps identifying the

[*]Equal contribution. X.shen is now at Amazon Alexa AI.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 325–330,
Aberdeen, Scotland, UK, 20-24 September 2021.

most informative instances to get labeled. We can consider the task as a starting step before applying active learning, after which more annotations can be continuously collected to further improve the model.

2 Task Description

Following the training scenario shown in Figure 1, we denote the unlabeled data as $U_1, U_2, \ldots, U_n$ where n is the data size. Depending on the downstream text generation task, "data" can mean different types of inputs like unlabeled structured data, documents and paragraphs respectively in the context of data-to-text, document summarization and question generation. We will select K instances from the whole unlabeled dataset, annotate them with reference text, and then train a neural generative model based on the annotated data. K is defined based on the annotation budget. In this work, since we focus on the few-shot scenario, K is set to be small ($\leq$ 2000). The goal is to *find the most representative K instances that can lead to the optimal performance when being annotated and trained on them*.

2.1 Submission Requirement

Participants are required to submit:

- An executable code that takes as input a set of unlabeled data, outputs K selected data that should be annotated.
- Selected training instances for $K = 50, 200, 500$ and 2000 together with model generations on the testset.
- A report that explains how the proposed selection strategy works and an analysis of its performance on the provided datasets.

While it is acceptable to take into account task or language specific features, participants are encouraged to submit selection strategies that are:

- Task agnostic. The selection strategy would work for a broad range of text generation tasks with various input-output formats.
- Language agnostic. The selection strategy can be seamlessly applied to same tasks in other languages.
- Model agnostic. The selection strategy can select most informative instances that improve the performance for a broad types of generative models (with various model architectures and training algorithms).
- K-agnostic. The selection strategy should work by varying the number of K.

2.2 Data

We will select representative datasets which cover three different text generation tasks. It will include but not limited to:

1. Data-to-text: We use the dataset for the E2E challenge (Novikova et al., 2017) which contain 50,602 data-text pairs with 8 unique slots in the restaurant domain.
2. Document Summarization: We use the CNN/Dailymail dataset (non-anonymized version) (Hermann et al., 2015) which contains 312,084 document-summary pairs.
3. Question generation: We use the SQuAD dataset (Rajpurkar et al., 2016) with over 100k questions. Following Du et al. (2017), we focus on the answer-independent scenario to directly generate questions from passages.

All the above datasets contain parallel input-output pairs for train/validation/test. We can simulate our few-shot scenario by only *allowing leveraging K input-output pairs from the training set*. The participants can decide which K training instances to select based on all the inputs in the training set [1]. Once the selected instances are determined, the model can then be trained on the K input-output pairs. It is also worth mentioning that in order to simulate the true few-shot scenario, *participants can only rely on the K input-output pairs for both training and validation*, i.e., no extra held-out examples are available for hyperparameter tuning and model selection (Schick and Schütze, 2020a; Perez et al., 2021). The participants can deside how to split them into the training and validation set.

We select the above three datasets only as examples for demonstration. Participants are encouraged to test their model on more diverse types of text generation tasks, e.g., tasks from the GEM benchmark (Gehrmann et al., 2021). Nevertheless, we recommend participants to first test and analyze

[1] The submitted instance selection algorithm can only condition on the inputs in the training set. However, participants are welcome to incorporate the reference text or testset distribution to analyze the theoretical upper bound performance.

their model on the above three datasets. In the final test, we will evaluate on the above three datasets to allow comparison across different submission. It is, however, totally acceptable to not target at all of the above three tasks. The participants can decide the tasks and datasets depending on their interest.

2.3 Generative Model

It is encouraged that participants can test their selection strategy on a wide list of generative models. In the end, to allow for a fair comparison across all submissions, we will test the selection algorithm by finetuning the open-sourced Bart model (Lewis et al., 2020) on the selected training instances with maximum likelihood. Bart is pretrained with a denoising autoencoder objective on large amount of text data and has been the state-of-the-arts for many text generation tasks. Therefore, we recommend to first test with this standard generative model. There have been many algorithms proposed for improved generation quality under the few-shot scenario like pattern exploitation training (Schick and Schütze, 2020a; Li and Liang, 2021; Lester et al., 2021) and cyclic training (Tseng et al., 2020; Chang et al., 2021a; Guo et al., 2021). We welcome test results using different types of generative models. Nonetheless, *the focus of the shared task is on the instance selection algorithm but not the few-shot generative model*. While it is nice to provide data points that demonstrate state-of-the-art results, generating with the most advanced model for better evaluation scores is by not means the main purpose.

2.4 Schedule

We follow the following schedule for the shared task of training instance selection:

- **December 15th, 2021**. The shared task is announced along with the selected text generation tasks and datasets.
- **February 15th, 2022**. The submission system and public leaderboard are open. Participants can deploy and test models with the provided automatic evaluation scripts.
- **May 15th, 2022**. This is the deadline for software and report submission. The manual evaluation begins. We will test the submitted selection algorithms with the same generative model and hyperparameter tuning mechanism. Model outputs will be compared with both automatic metrics and human evaluation.
- **June 15th, 2022**. The results of the automatic metrics and human evaluations will be announced.

After getting all the evaluation results, we will make a report to analyze different submissions. The shared task's findings are then presented at the following INLG.

3 Evaluation

The final evaluation will be conducted on the following two settings:

1. We apply the submitted selection algorithms to select K training instances and then finetune on them using a fixed strategy (with Bart model, same train/validation split and hyperparameter tuning mechanisms). The purpose is to evaluate all selection algorithms under a fair setting. In this setting, we will run the selection algorithm and training pipeline on our side to ensure fairness.

2. For each submission, we evaluate the model outputs of the best system. The purpose is to get an upper bound score for few-shot text generation with the best combination of settings (random seed, generative model, optimization algorithm, train/validation split, hyperparameter tuning, etc). In this setting, we will rely on the submissions of model outputs from the participants.

We will provide scripts for the automatic evaluation. The human evaluation will be conducted after all submissions are received under the same platform and metrics.

3.1 Automatic Evaluation

The evaluation metrics differ according to the downstream tasks. The metrics used for the final evaluation will be announced after the submission system is open. Participants are encouraged not to focus on one specific metric to avoid overfitting to it. The final evaluation will adopt metrics following into the following categories:

- Lexical similarity, which measure the lexical overlap between the model output and the gold references, including many popular metrics like BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and METEOR (Banerjee and Lavie, 2005).

- Semantic relevance, which measures the semantic similarity between the model output and the gold references, including the newly proposed BertScore (Zhang et al., 2019) and BLEURT (Sellam et al., 2020).

- Consistency with task input, which measures if the output contains consistent information with the task input and no hallucinations. Many works have proposed metrics based on question answering (Eyal et al., 2019; Durmus et al., 2020), natural language inference (Kumar and Talukdar, 2020) and mutual information (Shen et al., 2018; Zhang et al., 2018).

- Output diversity, which measures if the model can produce diverse outputs with different inputs, including metrics like the count and entropy of distinct uni/bi-grams (Li et al., 2016; Dušek et al., 2020).

- Other task-specific requirement, e.g., slot-error rate for data-to-text and compression rate for document summarization.

After the submission system opens, we will announce the metrics we picked for the automatic evaluation and provide the evaluation script.

3.2 Human Evaluation

We will also provide human evaluation scores on the system outputs since none of the automatic metrics can correlate perfectly with the generation quality. We will follow the recently proposed taxonomy of human evaluation measures by Belz et al. (2020); Su et al. (2020) and follow the reporting strategies proposed by Howcroft et al. (2020). The human evaluation will be focused on the following two parts, which are specifically hard to be accurately measured by automatic metrics:

- Fluency. If the output itself is a fluent sentence that can be well understood by humans, defined by a 5-scale Likert score.

- Consistency. If the output is consistent with the input and does not contain hallucinations, defined by a binary true/false score.

The human evaluation will be conducted after collecting all the submissions. It will be performed under a unified pipeline and annotation guideline to make sure results are comparable across model outputs from all submitted systems. To make the analysis comprehensive, participants are nonetheless encouraged to also perform their own human evaluation and include the results in their report.

3.3 Variance of Model

An important factor worth mentioning is the variance of the model. The variance of the model output can come from different steps, e.g., variance of the selection algorithm, random seed of training, hyperparameter selection, etc. It is rather straightforward to simply apply a random sampling strategy to select the K training instances and find a relatively good selection choice by brute force. However, this is clearly against the purpose of the shared task. We aim to find out a *selection algorithm that can stably help us identify the most representative training instances* instead of only getting the instance set. Therefore, when doing the final evaluation, if the submitted selection algorithm is not deterministic, we will run the algorithm 5 times to get 5 different selection sets and aggregate the results. The variance of the evaluation will also be reported (For the setting 1 of evaluation). For setting 2, we rely on the participants themselves to provide the selected instance set and the model outputs. Participants must indicate clearly how the instance set is determined, e.g., whether they cherry-pick a best instance set by randomly running the algorithm for many times, or leverage other information like the reference text for other inputs, testset distribution, etc.

4 Conclusion

In this proposal, we target at the problem of training instance selection for few-shot text generation. Current research simply applies random sampling which has a large selection variance and can lead to suboptimal performance. The main goal of the task is to call for more attention on this largely under-explored problem, gather innovative ideas on proposing selection algorithms and provide a fair platform for comparison.

We believe our shared task can be an important supplement to the study of few-shot text generation, where most works focus solely on the generative algorithm while neglecting the training instance selection. Selection strategies proposed in this task can be used to better benchmark model performances for few-shot text generation. Importantly, the task was inspired by realistic industrial settings and requirements and will hopefully bene-

fit multiple areas of NLP research including human-in-the-loop learning and other active learning based research, where the resource and time constraints calls for the task to be performed.

Acknowledgements

This research was funded in part by the German Research Foundation (DFG) as part of SFB 248 "Foundations of Perspicuous Software Systems". We sincerely thank the anonymous reviewers for their insightful comments that helped us to improve this paper.

References

Maria-Florina Balcan, Andrei Broder, and Tong Zhang. 2007. Margin based active learning. In *International Conference on Computational Learning Theory*, pages 35–50. Springer.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Anya Belz, Simon Mille, and David M. Howcroft. 2020. Disentangling the properties of human evaluation methods: A classification system to support comparability, meta-evaluation and reproducibility testing. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 183–194, Dublin, Ireland. Association for Computational Linguistics.

Ernie Chang, Jeriah Caplinger, Alex Marin, Xiaoyu Shen, and Vera Demberg. 2020. Dart: A lightweight quality-suggestive data-to-text annotation tool. In *Proceedings of the 28th International Conference on Computational Linguistics: System Demonstrations*, pages 12–17.

Ernie Chang, Vera Demberg, and Alex Marin. 2021a. Jointly improving language understanding and generation with quality-weighted weak supervision of automatic labeling. *Proceedings of EACL 2021*.

Ernie Chang, Xiaoyu Shen, Dawei Zhu, Vera Demberg, and Hui Su. 2021b. Neural data-to-text generation with lm-based text augmentation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 758–768.

Ernie Chang, Hui-Syuan Yeh, and Vera Demberg. 2021c. Does the order of training samples matter? improving neural data-to-text generation with curriculum learning. *Proceedings of EACL 2021*.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2020. Few-shot nlg with pre-trained language model. *ACL*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Association for Computational Linguistics (ACL)*.

Esin Durmus, He He, and Mona Diab. 2020. Feqa: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language*, 59:123–156.

Matan Eyal, Tal Baumel, and Michael Elhadad. 2019. Question answering as an automatic evaluation metric for news article summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3938–3948.

Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna Clinciu, Dipanjan Das, Kaustubh D Dhole, et al. 2021. The gem benchmark: Natural language generation, its evaluation and metrics. *arXiv preprint arXiv:2102.01672*.

Qipeng Guo, Zhijing Jin, Ziyu Wang, Xipeng Qiu, Weinan Zhang, Jun Zhu, Zheng Zhang, and Wipf David. 2021. Fork or fail: Cycle-consistent training with many-to-one mappings. In *International Conference on Artificial Intelligence and Statistics*, pages 1828–1836. PMLR.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

David M Howcroft, Anja Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: Nlg needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks. *arXiv preprint arXiv:2005.10433*.

Sawan Kumar and Partha Talukdar. 2020. Nile: Natural language inference with faithful natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8730–8742.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *arXiv preprint arXiv:2105.11447*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Timo Schick and Hinrich Schütze. 2020a. Few-shot text generation with pattern-exploiting training. *arXiv preprint arXiv:2012.11926*.

Timo Schick and Hinrich Schütze. 2020b. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892.

Xiaoyu Shen, Hui Su, Wenjie Li, and Dietrich Klakow. 2018. Nexus network: Connecting the preceding and the following in dialogue generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4316–4327.

Hui Su, Xiaoyu Shen, Zhou Xiao, Zheng Zhang, Ernie Chang, Cheng Zhang, Cheng Niu, and Jie Zhou. 2020. Moviechats: Chat like humans in a closed domain. In *Proceedings of EMNLP 2020*, pages 6605–6619.

Bo-Hsiang Tseng, Jianpeng Cheng, Yimai Fang, and David Vandyke. 2020. A generative model for joint natural language understanding and generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1795–1807.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 1815–1825.

Affective Decoding for Empathetic Response Generation

Chengkun Zeng[♠], **Guanyi Chen**[♡], **Chenghua Lin**[♠*], **Ruizhe Li**[♠], **Zhigang Chen**[♣]
[♠]Department of Computer Science, University of Sheffield
[♡]Department of Information and Computing Sciences, Utrecht University
[♣]College of Information and Intelligent Engineering, Zhejiang Wanli University
chengkunzenggo@gmail.com, {c.lin, r.li}@sheffield.ac.uk
g.chen@uu.nl, chenzhigang@zwu.edu.cn

Abstract

Understanding speaker's feelings and producing appropriate responses with emotion connection is a key communicative skill for empathetic dialogue systems. In this paper, we propose a simple technique called Affective Decoding for empathetic response generation. Our method can effectively incorporate emotion signals during each decoding step, and can additionally be augmented with an auxiliary dual emotion encoder, which learns separate embeddings for the speaker and listener given the emotion base of the dialogue. Extensive empirical studies show that our models are perceived to be more empathetic by human evaluations, in comparison to several strong mainstream methods for empathetic responding.

1 Introduction

Endowing a dialogue system with the ability of empathy responding has attracted a growing body of research (Ma et al., 2020) and is believed to be crucial for many service-oriented applications, such as mental health interventions (Hoermann et al., 2017), assisting medical diagnosis (Xu et al., 2019), and restaurant/hotel booking services (Ghazvininejad et al., 2017; Liu et al., 2018; Wang et al., 2021). Being empathetic requires one to be able to understand the implied feeling of the conversation partner, or in other words, to place oneself in partner's position. Therefore, to produce proper responses, an Empathetic Dialogue System (EDS) needs to understand not only the situation of the speaker[1] and the causes (Abd Yusof et al., 2017), but also the emotion state of speaker.

In 2019, Rashkin et al. (2019) formally introduced the task for dialogue systems to respond to conversations with emotions. They also constructed a benchmark corpus called EMPATHETICDIALOGUES (abbreviated as EMPDIAL), which consists of conversations with a wide range of emotion states for task evaluation. Figure 1 shows an example session of a dialogue from EMPDIAL, where the situation reflects the emotion state of *lonely* and *joyful*. Several approaches have been proposed for modelling emotions, which is a key challenge for building an EDS. These approaches follow two main enterprises: one is multi-task learning (Rashkin et al., 2019; Lin et al., 2020), which trains models for both dialogue generation and predicting the emotion of the dialogue; the other enforces the model to generate empathetic responses conditioning on the emotion state predicted from the dialogue context with a pre-trained

Figure 1: Two example dialogue sessions from the EMPDIAL dataset with the emotion *lonely* and *joyful*, repsectively.

[*]Corresponding author

[1]We use the term "speaker" referring to the user of the empathetic dialogue system while "listener" inferring to the dialogue system itself.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 331–340,
Aberdeen, Scotland, UK, 20-24 September 2021.

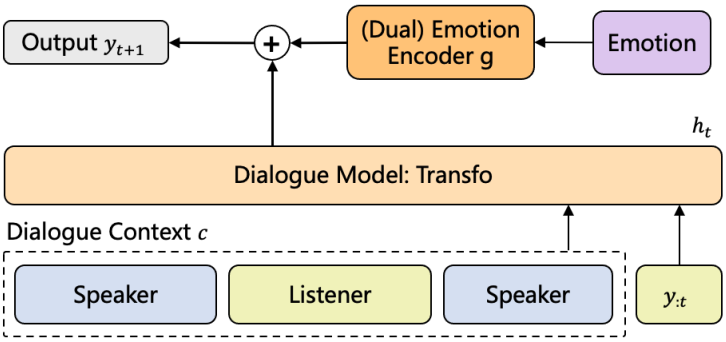


Figure 2: The overall architecture of our empathetic dialogue system.

emotion classifier (Rashkin et al., 2019).

Our work follows a similar vein to the second enterprise, where we propose a simple yet efficient technique coined *Affective Decoding* (AD), which can effectively incorporate emotion signals into model training and generate more empathetic responses. Our method can work in two different modes. The first mode injects emotion embedding in each decoding step. This is different to Rashkin et al. (2019) who only applied prepending at the first time-step on the encoder. For the second mode, we introduce an additional auxiliary *Dual Emotion Encoder*, which learns separate embeddings for the speaker and listener given the emotion base of the dialogue. In addition, we systematically evaluate and compare AD with the existing mainstream emotion modelling methods for empathetic responding, including both prepending emotion embeddings (Rashkin et al., 2019) and multi-task learning (Lin et al., 2020).

Based on the EMPDIAL dataset, we conducted comprehensive empirical studies, which include automatic (e.g., BLEU, BOW) and two human evaluations. For human evaluation, we assess both model-level performance and finer-grained level aspects concerning empathy, relevance, and fluency of the generated responses. Empirical results show the effectiveness of our affective decoding and that our model with the auxiliary dual emotion encoder works the best. While Rashkin et al. (2019) reported that multi-task learning did not provide consistent improvements for the task, we actually found multi-task learning performs even worse than a pre-trained language model (Wolf et al., 2019) fine-turned on the EMPDIAL dataset.

To summarise, our contributions are 3-fold: (1) we introduce a simple yet efficient decoding method called affective decoding to the task of empathetic response generation; (2) we conduct a comprehensive comparison between various emotion modelling methods in empathetic dialogue modelling by means of automatic evaluation and 2 human evaluations; (3) empirical results show the effectiveness of our affective decoding method and that with the auxiliary dual emotion encoder, our model can further support the analysis of listeners and speakers' behaviours in terms of how they utter with respected to the same emotion.

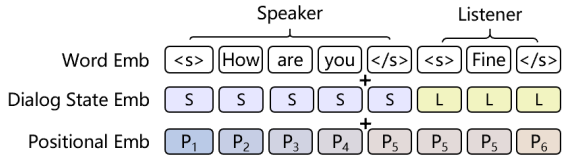


Figure 3: Illustration of the input format of our model.

The rest of the paper is organised as follows. §2 presents our model for empathetic response generation. We show the experimental setup and results in §3. §4 presents some case studies and finally §5 concludes the paper. The code is available at: https://github.com/zenggo/affective-decoding-4-empathetic-dialog.

2 Methodology

In this section, we describe our Affective Decoding model, which consists of two key components, namely, a pre-trained response generator, *Transfo* (Wolf et al., 2019), and the affective decoder for enhancing empathetic responding.

2.1 Dialogue Modelling

We use Transfo, which is built upon the Generative Pre-trained Transformer (Radford, 2018, GPT) pre-trained on the BOOKSCORPUS dataset (Zhu et al., 2015), and which gives good performance on

building conversational agents (Dinan et al., 2020). When fine-tuning on the EMPDIAL dataset, a response is generated given the dialogue context c, which contains single or multi-turn conversations. For each input token, it is represented as a summation of its word embedding, positional embedding and dialogue state embedding, as illustrated in Figure 3. We model two possible dialogue states, where state S corresponds to the speaker and state L to the listener.

2.2 Affective Decoding

One of the key challenges of building an effective EDS is recognising and understanding the emotion of the speaker. Inspired by the affective language model of Ghosh et al. (2017), we tackle the problem by proposing *Affective Decoding* (**AD**), a simple strategy which injects emotion embeddings into each decoding step. Such a strategy allows our model to encode dialogue's emotion base effectively, and to distribute more probability mass towards the words in the utterance that are highly correlated with the dialogue emotion, leading to enhanced empathetic responding.

Concretely, at each time step t, we first encode the emotional label with a one-hot embedding, which is then mapped into a dense vector $g(\mathbf{e})$ by the emotion encoder $g(\cdot)$ (see Figure 2 for details). $g(\mathbf{e})$ is then used for predicting the next word y_{t+1} jointly with the dialogue context c and the decoded outputs for all previous time steps $y_{:t}$. Formally, the probability of $P(y_{t+1})$ is given as

$$P(y_{t+1}|y_{:t}, c, e) = \text{softmax}(Wh_t + Vg(\mathbf{e})), \quad (1)$$

where h_t is the representation of c and $y_{:t}$ encoded by *Transfo*; W and V are weights in the output layer. Similar to prior studies (Rashkin et al., 2019; Lin et al., 2020), our AD model maintains one emotion embedding for the whole dialogue session.

Dual Emotion encoder. We observe that in the dialogues with emotional situations, speaker and listener tend to utter with distinctive styles. That is, the speaker normally describes his/her own experience with personal emotions, while the listener tries to respond in the way which can establish an emotional connection with the speaker based on speaker's emotional needs (e.g., encouraging and motivating). For example, in the dialogue with a emotional base of *joyful* in Figure 1, the speaker used phrases like *"happiness"* and *"love"* while listeners used *"exciting"* and *"congratulation"*. Based on this observation, we introduce a mechanism so called Dual Emotion (DE) encoder, which learns separate embeddings for the speaker and listener given the emotion base of the dialogue. We coin our model augmented with the auxiliary DE component **AD+DE**, and its generation process becomes:

$$P(y_{t+1}|y_{:t}, c, e) = \begin{cases} \text{softmax}(Wh_t + V_S g_S(\mathbf{e})) & s_t = S \\ \text{softmax}(Wh_t + V_L g_L(\mathbf{e})) & s_t = L, \end{cases} \quad (2)$$

where $s_t \in \{S, L\}$ is the dialogue state at the step t. With the dual embedding space, we hypothesise that the interpretability of our model's behaviour will also be enhanced, as it makes possible to identify the differences of the language use between speakers and listeners.

3 Experiment

We evaluate our models on EMPDIAL, using the original split of Rashkin et al. (2019) and their emotion classifier based on FastText (Joulin et al., 2017). Table 1 shows the statistics of the EMPDIAL dataset, which contains 32 emotion labels. We compare our model against four competitive baselines in the experiment, including

- **MoEL**: a transformer model with multiple independent transformer decoders for generating different contextual responses (Lin et al., 2019);
- **Transfo**: a pre-trained transformer model which is fine-tuned using multi-task learning in language modelling and next-utterance classification tasks (Wolf et al., 2019);
- **PRE**: a Transfo model with an emotion embedding prepended to the dialogue context (Rashkin et al., 2019);
- **MTL**: a Transfo model with multi-task learning, where the main task is dialogue response generation and the secondary task uses the encoded dialogue context for predicting the emotion for the whole session (Lin et al., 2020).

In terms of our own models, apart from **AD** and **AD+DE**, we also further tested a model variant (**ADM**), which considers multi-task learning. We detail each of our model variants below.

- **AD**: a simple model by injecting emotion embeddings into each decoding step;

Dataset	Emotion Labels	Train	Validation	Test
EMPDIAL	32	19,533	2,770	2,547

Table 1: The statistics of EMPDIAL dataset.

Model	PPL	BLEU	BOW_e	BOW_a	BOW_g	DIST-1	DIST-2
MoEL	38.19	**2.84**	0.502	**0.878**	**0.679**	0.005	0.023
Transfo	13.94	1.75	0.729	0.747	0.559	0.015	0.070
Prepend	**13.90**	1.75	0.731	0.753	0.565	**0.016**	**0.079**
MTL	14.95	1.49	0.733	0.757	0.566	0.015	0.067
ADM	14.50	1.81	0.733	0.756	0.564	0.015	0.068
AD	14.04	1.69	0.734	0.756	0.570	**0.016**	0.072
AD+DE	14.03	1.71	**0.736**	0.757	0.571	**0.016**	0.074

Table 2: Automatic evaluation results.

- **AD+DE**: a variant of **AD** by introducing the Dual Emotion encoder to separately model embeddings for the speaker and listener;
- **ADM**: a variant of our model by combining **AD+DE** with multi-task learning, adopting a similar strategy to the **MTL** baseline.

3.1 Automatic Evaluation

For automatic evaluation, we evaluate the models in three aspects, i.e., *fluency*, *adequacy*, and *diversity*. Particularly, fluency is measured by perplexity, adequacy by BLEU and BOW embedding metrics, and diversity by DIST. We describe each of the metrics in detail below.

- **Perplexity (PPL)**: measures how well a language model is (lower the better);
- **BLEU** (Papineni et al., 2002): n-gram overlap between the system output and the reference;
  BOW embedding (Liu et al., 2016b): the cosine similarity between the bag-of-words embeddings of the output and the reference. Specifically, there are three matching strategies:
  - Greedy (**BOW_g**): the average cosine similarities between word embeddings of the two utterances which are greedily matched (Rus and Lintean, 2012);
  - Average (**BOW_a**): the cosine similarity between the averaged word embeddings in the two utterances (Mitchell and Lapata, 2008);
  - Extreme (**BOW_e**): the cosine similarity between the largest extreme values in the word embeddings of the two utterances (Pennington et al., 2014);
- **DIST** (Li et al., 2016): measures the corpus level diversity of the outputs by calculating the ratio of unique n-grams ($n = 1, 2$) over all n-grams in the outputs.

3.1.1 Experimental Results

Table 2 shows the automatic evaluation results for the tested models. Overall, the results do not seem to provide strong evidence in terms of which models perform best. Among the baselines, MoEL achieves the highest BLEU, BOW_a and BOW_g, while it has the worst PPL and diversity (i.e., DIST-1 and DIST-2). Prepend in contrast, performs the best in terms of PPL and diversity, and gives similar performance in the BOW metrics when compared to other baselines (except MoEL) and our models.

Our AD+DE model gives similar performance to Prepend, i.e., it achieves fairly balanced performance across all types of metrics and gives the highest scores in BOW_e and DIST-1. AD+DE also appears to slightly outperform AD, but the difference is somewhat minimal. In addition, it is surprising to see no significant difference between Transfo and other models for all metrics, where the latter explicit account for the emotional signals of the dialogue. We also see that MTL has a lower BLEU score even than Transfo. Conversely, comparing AD+DE and ADM, multi-task learning helps to yield better BLEU but yields worse performance

on other metrics. In summary, we are not able to establish a clear winner based on automatic metrics, although Prepend seems to slightly outperform other baseline models overall.

3.2 Human Evaluation

To assess the performance of the tested models more robustly and comprehensively, we conducted two forms of human evaluation: *ranking* for evaluating the overall performance of each system (Duh, 2008), and *multi-item rating* (Diamantopoulos et al., 2012) for evaluating the system performance against more fine-grained aspects (e.g. whether the response is relevant or not).

3.2.1 Ranking based Human Evaluation

We use pairwise binary ranking (i.e., preference test) (Vilar et al., 2007), which has been shown reliable for comparing the performance of multiple models. We randomly sample 100 dialogue context from the test set for both single-turn and multi-turn dialogues (i.e., 50 samples each type). We then generate a response with each tested model given a sampled context. Given two responses generated by two models, two raters (PhD students in computer science) were asked to decide *which model is better in terms of empathetic responding* or *there is no difference*.

We report the results of this pairwise preference test in Table 3, and the corresponding break down results of the single-turn and multi-turn dialogues in Table 4. Take the number 48 corresponding to Transfo and MoEL in Table 3 as an example. It means that 48% of the judges prefer Transfer over MoEL by considering both single-turn and multi-turn dialogues in the test set. Table 4 gives the break down results for single-turn (i.e., 46%) and multi-turn dialogue (i.e., 50%), respectively. By taking the average, we can derive 48% as the overall result. Clearly, human evaluation (i.e., Table 3) shows very different observations compared to the automatic evaluation. On the one hand, AD and AD+DE are clear winners this time, which significantly outperform all other models including the best performed baseline *PRE*. It can also be observed that AD+DE slightly outperformed AD but the difference is insignificant. On the other hand, multi-task learning shows a negative effect on empathetic dialogue modelling, i.e., by comparing MTL with Transfo and by comparing ADM with AD+DE. We give more discussions regarding this phenomenon in the Rating experiment section.

In addition, it can be observed that MoEL gives the worst performance compared to all other models by a large margin, but one might argue that the results are not directly comparable because the non pre-trained MoEL has less capacity than other pre-trained baseline models (e.g., the parameters of Transfo are 5 times as many as that of MoEL). The inconsistency on the results of automatic evaluation and preference test somewhat resemble the observation of prior studies that automatic metrics show low validity for evaluating empathetic dialogue systems (Liu et al., 2016a). To further investigate the underlying issue, we interview our raters as to which factor influences most on their decisions. It turns out that small errors in the responses that cannot be detected by the automatic measures (e.g. BLEU or BOW) can have a great impact. For instance, wrong reference (e.g., responding *"I'm happy for you."* when the speaker is actually describing an experience of his/her sister) or wrong tense (e.g., responding *"I hope you will be fine."* when the speaker is describing an experience happened in the past).

3.2.2 Rating based Human Evaluation

Likert Scale Rating (LSR) and Magnitude Estimation (ME) are two popular rating based methods. It is reported that ME performs better for evaluating goal-oriented dialogue systems (Santhanam and Shaikh, 2019) and language generation systems (Novikova et al., 2018) while LSR works better for measuring acceptability of text (Langsford et al., 2018). Considering the degree of empathetic is tied to the acceptability of the generated responses and that multi-item LSR is on a par with ME (van der Lee et al., 2019), we opt for LSR with three dimensions listed below. Model responses (the same set used in the ranking study) were scored by same two raters. The rating score ranges from 0 to 3.

- Empathy: *Does the listener understand the speaker's feelings, and responds appropriately?*
- Relevance: *Is the content of the reply relevant to the topic mentioned by the speaker? Is it informative?*
- Fluency: *Does the response look fluent?*

The rating results in Table 5 and Table 6 (break down results) show a similar tendency with the

Model	MoEL	Transfo	PRE	MTL	ADM	AD	AD+DE
MoEL	-	12	11	26	18	11	12
Transfo	48†	-	23	28	29	13	17
PRE	52†	34†	-	49†	34	19	22
MTL	48†	25	19	-	27	16	15
ADM	56†	45†	26	46†	-	22	21
AD	55†	39†	45†	44†	40†	-	8
AD+DE	57†	42†	35†	45†	43†	12	-

Table 3: Results of the pairwise preference test: the number indicates the percentage (%) of responses generated by system A (row) is favoured by raters comparing to B (column). † means the preference is significant with $p < .05$ (two-proportion z-test).

		Single-turn							Multi-turn						
	Model	1	2	3	4	5	6	7	1	2	3	4	5	6	7
1	MoEL	-	15	12	29	19	14	16	-	9	10	23	17	8	8
2	Transfo	46	-	14	28	36	14	22	50	-	32	28	22	12	12
3	Prepend	47	42	-	46	44	24	20	57	26	-	52	24	14	24
4	MTL	42	32	16	-	26	16	16	54	18	22	-	28	16	14
5	MTL+AD+DE	52	44	22	44	-	28	28	60	46	30	48	-	16	14
6	AD	49	52	40	48	46	-	16	61	26	50	40	34	-	8
7	AD+DE	51	52	32	42	48	10	-	63	32	38	48	38	6	-

Table 4: Results of the pairwise preference test for single-turn EDS (left) and multi-turn EDS (right): the number indicates the percentage (%) of responses generated by system A (row) is favoured by raters comparing to B (column).

Model	Empathy	Relevance	Fluency
MoEL	1.629†‡	1.525†‡	2.031†‡
Transfo	1.987†	2.043†	2.307
PRE	1.940†	2.043†	2.270
MTL	1.850†	1.917†	2.313
ADM (ours)	2.020†	2.012†	2.330
AD (ours)	2.177‡	2.233‡	2.380
AD+DE (ours)	2.187‡	2.237‡	2.387

Table 5: LSR Results. † means AD+DE significantly outperforms the corresponding model with $p < .05$ and ‡ means the corresponding model outperforms Transfo significantly with $p < .05$ (paired t-test).

ranking experiment and give some additional insights. Regarding *fluency*, how or whether a model incorporates emotion information seems to have no impact to the fluency of the generated responses. In terms of the other two aspects, we have the following findings: (1) similar to the ranking experiment, MoEL gives the worst performance, regardless the rating aspect. Comparing to *transfo*, the PRE, MTL, and ADM models cannot improved either *empathy* or *relevance* of the generated responses, significantly. The remarkable well performance of the vanila *transfo* model embodies that by fine-turning the model on EMPDIAL, this GPT based model is a decent baseline for understanding emotion and responding empathetically; (2) in line with the ranking experiment, AD and AD+DE give the best performance. Although AD+DE performs slightly better than AD, the difference between them is not significant. Joining with other results, it seems that learning separate embeddings for the speaker and listener does bring some benefit but it is not as strong as expected. Nonetheless, we found that introducing DE can help analyse the behaviours of listeners and speakers in terms of how they utter with respected to the same emotion situation, which will be discussed in detail in §4.2; (3) comparing the results of MTL with Transfo and that of ADM with AD+DE reveals that MTL decreases both empathy and relevance of the responses. One possible reason behind why MTL does not yield positive effect in EDS (based on the results of both ranking and rating experiments) is that there might exist trade-off between the optimisation of the dialogue generator's objective and that of the emotion classifier's objective (Sener and Koltun, 2018). As a result, the overall performance is harmed by the naive linear combination of the two objectives.

	Single-turn			Multi-turn		
Model	Empathy	Relevance	Fluency	Empathy	Relevance	Fluency
MoEL	1.714	1.652	2.050	1.543	1.397	2.012
Transfo	2.073	2.193	2.373	1.900	1.893	2.240
Prepend	1.920	2.047	2.327	1.960	2.040	2.213
MTL	1.920	2.027	2.353	1.780	1.807	2.273
MTL+AD+DE	2.067	2.093	2.340	1.973	1.933	2.320
AD	2.187	2.260	2.373	2.187	2.213	2.400
AD+DE	2.180	2.240	2.367	2.173	2.227	2.393

Table 6: LSR results for single-turn and multi-turn dailogues respectively.

	Generated Response
Input	S: I got a free pizza from Domino's but felt bad about how it happened.
Transfo	oh no ! what did you do ?
PRE	oh no , what happened ?
MTL	i love domino 's pizza ! i love domino 's !
ADM	oh no , what happened ?
AD	why did you feel bad about it ?
AD+DE	why did you feel bad about it ?
Input	S: Do any of your kids play any sports? L: No, I do not have kids. Yours? S: My daughter skateboards. She went down the big ramp the other day. It was so cool.
Transfo	that sounds like a lot of fun !
PRE	wow , that must have been a lot of fun .
MTL	that 's cool . i 've never been on a rollercoaster . i 've never been on a rollercoaster .
ADM	that is so cool ! i bet you are so proud of her !
AD	that 's awesome ! i bet you were so proud !
AD+DE	that 's awesome ! i bet you were so proud of her !

Table 7: Example outputs generated by the baselines and our models, where S stands for speaker and L stands for Listener.

4 Case Studies

4.1 Sample Outputs of Different Empathetic Dialogue Systems

Table 7 lists a number of sample responses generated by the baselines and our models. It can be observed that our AD and AD+DE models produce high quality empathetic responses. For the first example, our models can follow the context and ask the reason why the speaker felt bad about getting a free pizza, whereas some of the baseline models produce uninformative responses (e.g., *what did you do?*) and some of them respond with incorrect emotion (e.g., *I love domino's pizza!*). In the second sample, our models can generate more empathetic responses (i.e., providing more approval and praise) compared to other baselines. In contrast, method like MTL generate irrelevant content (i.e., *rollercoaster*). Another observation is that the responses generated by AD and AD+DE are quite similar to each other, which is in line with the evaluation results.

4.2 Interpreting Dual Emotional Embeddings

We also conducted an experiment to assess how the learnt emotion embeddings by AD+DE differ with respect to speakers and listeners. Given an emotion label, we listed the label's top-10 nearest neighbours in the speaker space and listener

Emotion State	Nearest neighbour words of the emotion label
Proud	**S**: son, graduated, proud, honour, daughter, happy, pleased, nephew, musicians, said **L**: celebrate, bet, con, proud, keep, parent, started, moment, congratulations
Sad	**S**: sad, cried, upset, bummed, died, passed, cry, depressed **L**: sorry, retrace, memories, sleazy, lose, toll, alive, sudden

Table 8: The 10 nearest neighbour words of the emotion label **PROUD** and **SAD** in the speaker (S) and listener (L) space, respectively.

space (see Table 8), respectively, based on the label embedding. Take the emotion label "proud" as an example, words like *proud*, *happy*, *honour* in the speaker space are very close semantically and are highly relevant to the emotion label. Also words like *son*, *daughter* are often be mentioned in parents' expression of pride. In the listener space, words like *congratulations*, *proud*, *celebrate* are commonly used for responding to the speaker's emotion of proud and the corresponding experience. These examples not only show consistency with people's conversation habits, but also illustrate the difference between the speaker's and the listener's diction.

4.3 Generating Empathetic Dialogues from Scratch

Since we jointly model the speakers and listeners in the empathetic dialogues, our system is capable to generate a multi-turn conversation given an emotional situation and a prompt. Figure 4 provides some example dialogues generated by AD+DE in such a way. After given a specific emotion label (e.g., *joyful* and *disappointed* from the predefined label set of the EMPDIAL dataset), our model can generate relevant and empathetic responses conditioned on the initial prompt such as "my mother". It can be observed that the generated multi-turn conversations are coherent and respect the given emotion labels.

[joyful]
S: my mother just got a promotion at her job !
L: that 's great ! what kind of job is it ?
S: it 's a financial analyst job !
L: that 's great ! i 'm sure you 're proud of her !

[disappointed]
S: my mother was diagnosed with pan cre atic cancer a few weeks ago .
L: oh no ! i 'm sorry to hear that . is she going to be okay ?
S: i think so , but i was n't expecting it at all .
L: i 'm sorry to hear that . i hope everything works out for you .

Figure 4: Given the initial word *my mother*, two example dialogues are generated conditioning on the given emotion "joyful" and "disappointed".

5 Conclusion

In this paper, we propose a simple and effective technique called Affective Decoding for empathetic response generation. Empirical results based on extensive human evaluation show that our models (AD and AD+DE) outperform several strong baselines. Simply fine-tune the pre-trained *Transfo* on EMPDIAL achieves decent performance. MTL, which has been used in some EDS, shows negative effects on the overall performance. As a side outcome, we also confirm the low validity of the mainstream automatic metrics for evaluating empathetic dialogue systems.

It was noted that empathetic dialogue systems tend to generate generic responses such as "*I'm sorry to hear that.*". Therefore, one important future work is to improve the diversity and informativeness of the empathetic responses generated by an EDS. One possible technical direction is to employ variational autoenoders (Zhao et al., 2017; Li et al., 2019, 2020), which have been shown effective in improving the diversity in response generation.

Acknowledgement

This work is supported by the awards made by the UK Engineering and Physical Sciences Research Council (EP/P011829/1) and Ningbo Natural Science Foundation (202003N4320, 202003N4321). We thank anonymous reviewers for their insightful comments.

References

Noor Fazilla Abd Yusof, Chenghua Lin, and Frank Guerin. 2017. Analysing the causes of depressed mood from depression vulnerable individuals. In *Proceedings of the International Workshop on Digital Disease Detection using Social Media 2017 (DDDSM-2017)*, pages 9–17, Taipei, Taiwan. Association for Computational Linguistics.

Adamantios Diamantopoulos, Marko Sarstedt, Christoph Fuchs, Petra Wilczynski, and Sebastian Kaiser. 2012. Guidelines for choosing between multi-item and single-item scales for construct measurement: a predictive validity perspective. *Journal of the Academy of Marketing Science*, 40(3):434–449.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2020. The second conversational intelligence challenge (convai2). In *The NeurIPS'18 Competition*, pages 187–208. Springer.

Kevin Duh. 2008. Ranking vs. regression in machine translation evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 191–194.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2017. A knowledge-grounded neural conversation model. *arXiv preprint arXiv:1702.01932*.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis Philippe Morency, and Stefan Scherer. 2017. Affect-LM: A neural language model for customizable affective text generation. *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 1:634–642.

Simon Hoermann, Kathryn L McCabe, David N Milne, and Rafael A Calvo. 2017. Application of synchronous text-based dialogue systems in mental health interventions: systematic review. *JMIR*, 19(8):e267.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Steven Langsford, Amy Perfors, Andrew T Hendrickson, Lauren A Kennedy, and Danielle J Navarro. 2018. Quantifying sentence acceptability measures: Reliability, bias, and variability. *Glossa: a journal of general linguistics*, 3(1).

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368, Tokyo, Japan. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Ruizhe Li, Xiao Li, Guanyi Chen, and Chenghua Lin. 2020. Improving variational autoencoder for text modelling with timestep-wise regularisation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2381–2397, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ruizhe Li, Xiao Li, Chenghua Lin, Matthew Collinson, and Rui Mao. 2019. A stable variational autoencoder for text modelling. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 594–599, Tokyo, Japan. Association for Computational Linguistics.

Zhaojiang Lin, Andrea Madotto, Jamin Shin, Peng Xu, and Pascale Fung. 2019. Moel: Mixture of empathetic listeners. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 121–132.

Zhaojiang Lin, Peng Xu, Genta Indra Winata, Farhad Bin Siddique, Zihan Liu, Jamin Shin, and Pascale Fung. 2020. Caire: An end-to-end empathetic chatbot. In *AAAI*, pages 13622–13623.

Bing Liu, Gokhan Tür, Dilek Hakkani-Tür, Pararth Shah, and Larry Heck. 2018. Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2060–2069, New Orleans, Louisiana. Association for Computational Linguistics.

Chia Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016a. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 2122–2132.

Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016b. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Yukun Ma, Khanh Linh Nguyen, Frank Z Xing, and Erik Cambria. 2020. A survey on empathetic dialogue systems. *Information Fusion*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *proceedings of ACL-08: HLT*, pages 236–244.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2018. Rankme: Reliable human ratings for natural language generation. *arXiv preprint arXiv:1803.05928*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Alec Radford. 2018. Improving Language Understanding by Generative Pre-Training.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards Empathetic Open-domain Conversation Models: A New Benchmark and Dataset. pages 5370–5381.

Vasile Rus and Mihai Lintean. 2012. An optimal assessment of natural language student input using word-to-word similarity metrics. In *International Conference on Intelligent Tutoring Systems*, pages 675–676. Springer.

Sashank Santhanam and Samira Shaikh. 2019. Towards best experiment design for evaluating dialogue system output. *arXiv preprint arXiv:1909.10122*.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *arXiv preprint arXiv:1810.04650*.

David Vilar, Gregor Leusch, Hermann Ney, and Rafael E Banchs. 2007. Human evaluation of machine translation through binary system comparisons. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 96–103.

Dingmin Wang, Chenghua Lin, Qi Liu, and Kam-Fai Wong. 2021. Fast and scalable dialogue state tracking with explicit modular decomposition. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–295, Online. Association for Computational Linguistics.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents. *arXiv preprint arXiv:1901.08149*.

Lin Xu, Qixian Zhou, Ke Gong, Xiaodan Liang, Jianheng Tang, and Liang Lin. 2019. End-to-end knowledge-routed relational dialogue system for automatic diagnosis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7346–7353.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, Vancouver, Canada. Association for Computational Linguistics.

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

Controllable Sentence Simplification with a Unified Text-to-Text Transfer Transformer

Kim Cheng Sheang, Horacio Saggion
LaSTUS, TALN, Universitat Pompeu Fabra
C/Roc Boronat, 138, 08018 Barcelona, Spain
{kimcheng.sheang,horacio.saggion}@upf.edu

Abstract

Recently, a large pre-trained language model called T5 (A Unified Text-to-Text Transfer Transformer) has achieved state-of-the-art performance in many NLP tasks. However, no study has been found using this pre-trained model on Text Simplification. Therefore in this paper, we explore the use of T5 fine-tuning on Text Simplification combining with a controllable mechanism to regulate the system outputs that can help generate adapted text for different target audiences. Our experiments show that our model achieves remarkable results with gains of between +0.69 and +1.41 over the current state-of-the-art (BART+ACCESS). We argue that using a pre-trained model such as T5, trained on several tasks with large amounts of data, can help improve Text Simplification.[1]

1 Introduction

Text Simplification (TS) can be regarded as a natural language generation task where the generated text has a reduced language complexity in both vocabulary and sentence structure while preserving its original information and meaning (Saggion, 2017). Its applications can be used as reading assessment tools for people with low-literacy skills such as children (Watanabe et al., 2009), and non-native speakers (Paetzold and Specia, 2016), or people with cognitive disabilities such as autism (Barbu et al., 2015), aphasia (Carroll et al., 1999), and dyslexia (Rello et al., 2013a; Matausch and Peböck, 2010). In addition, TS can also be used as a preprocessing step to improve the results of many NLP tasks, e.g., Parsing (Chandrasekar et al., 1996), Information Extraction (Evans, 2011; Jonnalagadda and Gonzalez, 2010), Question Generation (Bernhard et al., 2012), Text Summarization (Siddharthan et al., 2004), and Machine Translation (Štajner and Popović, 2016, 2019).

In recent years, research in TS has been mostly focused on developing models based on deep neural networks (Vu et al., 2018; Zhao et al., 2018b; Martin et al., 2020b). However, and to the best of our knowledge, very few studies of transfer learning –where a model is first pre-trained on a data-rich task and then fine-tuned on downstream tasks– have been explored in TS.

In this paper, we propose a transfer learning and controllable sentence simplification model that harnesses the power of the Unified Text-to-Text Transfer Transformer (T5) pre-trained model (Raffel et al., 2020), combining it with control tokens to provide a way to generate output that adapts to different target users. Such a model can be adjusted to fit the need of different users without having to build everything from the ground up.

We make the following contributions:

- We introduce a transfer learning approach combined with a controllable mechanism for sentence simplification task.
- We make an improvement to the performance of the sentence simplification system.
- We introduce a new control token #words to help the model generate sentences by replacing long complex words with shorter alternatives.
- We conduct an evaluation and comparison between different sizes of pre-trained models and a detailed analysis on the effect of each control token.
- We show that by choosing the right control token values and pre-trained model, the model achieves the state-of-the-art performance in two well-known benchmarking datasets.

[1]The code and data are available at https://github.com/KimChengSHEANG/TS_T5

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 341–352,
Aberdeen, Scotland, UK, 20-24 September 2021.

2 Related Work

2.1 Sentence Simplification

It is often regarded as a monolingual translation problem (Zhu et al., 2010; Coster and Kauchak, 2011; Wubben et al., 2012), where the models are trained on parallel complex-simple sentences extracted from English Wikipedia and Simple English Wikipedia (SEW) (Zhu et al., 2010).

There are many approaches based on statistical Machine Translation (SMT), including phrase-based MT (PBMT) (Štajner et al., 2015), and syntax-based MT (SBMT) (Xu et al., 2016). Nisioi et al. (2017) introduced Neural Text Simplification (NTS), a Neural-Machine-Translation-based system (NMT) which performs better than SMT. Zhang and Lapata (2017) took a similar approach adding lexical constraints combining the NMT model with reinforcement learning. After the release of Transformer (Vaswani et al., 2017), Zhao et al. (2018a) introduced a Transformer-based approach and integrated it with a paraphrase database for simplification called Simple PPDB (Pavlick and Callison-Burch, 2016a). The model outperforms all previous state-of-the-art models in sentence simplification.

Our proposed model is also a sequence-to-sequence Transformer-based model, but instead of using the original Transformer by Vaswani et al. (2017), we use T5 (Raffel et al., 2020).

2.2 Controllable Sentence Simplification

In recent years, there has been increased interest in conditional training with sequence-to-sequence models. It has been applied to some NLP tasks such as controlling the length and content of summaries (Kikuchi et al., 2016; Fan et al., 2017), politeness in machine translation (Sennrich et al., 2016), and linguistic style in text generation (Ficler and Goldberg, 2017). Scarton and Specia (2018) introduced the controllable TS model by embedding grade level token <grade> into the sequence-to-sequence model. Martin et al. (2020b) took a similar approach adding 4 tokens into source sentences to control different aspects of the output such as length, paraphrasing, lexical complexity, and syntactic complexity. Kariuk and Karamshuk (2020) took the idea of using control tokens from Martin et al. (2020b) and used it in unsupervised approach by integrating those control tokens into the back translation algorithm, which allows the model to self-supervise the process of learning inter-relations between a control sequence and the complexity of the outputs. The results of Scarton and Specia (2018), Martin et al. (2020b), and Kariuk and Karamshuk (2020) have shown that adding control tokens does help improve the performance of sentence simplification models quite significantly.

Building upon Martin et al. (2020b), we fine-tune T5 with all control tokens as defined in Martin et al. (2020b) to control different aspects of the output sentences. Moreover, we add one more control token (number of words ratio) in order to be able to generate new sentences with a similar length as the source but shorter in word length as we believe that the number characters ratio alone is not enough for the model to generate shorter words.

3 Model

In this work, we fine-tune T5 pre-trained model with the controllable mechanism on Text Simplification. T5 (A Unified Text-to-Text Transfer Transformer) (Raffel et al., 2019) is pre-trained on a number of supervised and unsupervised tasks such as machine translation, document summarization, question answering, classification tasks, and reading comprehension, as well as BERT-style token and span masking (Devlin et al., 2019). There are five different variants of T5 pre-trained models: T5-small (5 attention modules, 60 million parameters), and T5-base (12 attention modules, 220 million parameters). Due to the limited resources of Colab Pro, we are able to train only T5-small and T5-base.

3.1 Control Tokens

We use control tokens to control different aspects of simplification such as compression ratio (#Chars), paraphrasing (Levenshtein similarity), lexical complexity (word rank), and syntactic complexity (the depth of dependency tree) as defined in (Martin et al., 2020b). Then, we add another control token word ratio (#Words) to control word length. We argue that word ratio is another important control token because normally word frequency correlates well with familiarity, and word length can be an additional factor as long words tend to be hard to read (Rello et al., 2013b). Moreover, corpus studies of original and simplified texts show that simple texts contain shorter and more frequent words (Drndarević and Saggion, 2012). Therefore, we add word ratio to help the model generate sim-

plified sentences with a similar amount of words and shorter in word length, whereas #Chars alone could help the model regulate sentence length but not word length.

- **#Chars (C):** character length ratio between source sentence and target sentence. The number of characters in target divided by that of the source.
- **LevSim (L):** normalized character-level Levenshtein similarity (Levenshtein, 1966) between the source and target.
- **WordRank (WR):** inverse frequency order of all words in the target divided by that of the source.
- **DepTreeDepth (DTD):** maximum depth of the dependency tree of the target divided by that of the source.
- **#Words (W):** number of words ratio between source sentence and target sentence. The number of words in target divided by that of the source.

Table 1 shows an example of a sentence embedded with control tokens for training.

Source
simplify: W_0.58 C_0.52 L_0.67 WR_0.92 DTD_0.71 In architectural decoration Small pieces of colored and iridescent shell have been used to create mosaics and inlays, which have been used to decorate walls, furniture and boxes.
Target
Small pieces of colored and shiny shell has been used to decorate walls, furniture and boxes.

Table 1: This table shows how control tokens are embedded into the source sentence for training. The keyword **simplify** is added at the beginning of each source sentence to mark it as a simplification task.

4 Experiments

Our model is developed using the Huggingface Transformers library (Wolf et al., 2019)[2] with PyTorch[3] and Pytorch lightning[4].

[2] https://huggingface.co/transformers/model_doc/t5.html

[3] https://pytorch.org

[4] https://pytorchlightning.ai

4.1 Datasets

We use the WikiLarge dataset (Zhang and Lapata, 2017) for training. It is the largest and most commonly used text simplification dataset containing 296,402 sentence pairs from automatically aligned complex-simple sentence pairs English Wikipedia and Simple English Wikipedia which is compiled from (Zhu et al., 2010; Woodsend and Lapata, 2011; Kauchak, 2013).

For validation and testing, we use TurkCorpus (Xu et al., 2016), which has 2000 samples for validation and 359 samples for testing, and each complex sentence has 8 human simplifications. We also use a newly created dataset called ASSET (Alva-Manchego et al., 2020) for testing, which contains 2000/359 samples (validation/test) with 10 simplifications per source sentence.

4.2 Evaluation Metrics

Following previous research (Zhang and Lapata, 2017; Martin et al., 2020a), we use automatic evaluation metrics widely used in text simplification task.

SARI (Xu et al., 2016) compares system outputs with the references and the source sentence. It measures the performance of text simplification on a lexical level by explicitly measuring the goodness of words that are added, deleted and kept. So far, it is the most commonly adopted metric and we use it as an overall score.

BLEU (Papineni et al., 2002) is originally designed for Machine Translation and is commonly used previously. BLEU has lost its popularity on Text Simplification due to the fact that it correlates poorly with human judgments and often penalizes simpler sentences (Sulem et al., 2018). We keep using it so that we can compare our system with previous systems.

FKGL (Kincaid et al., 1975) In addition to SARI and BLEU, we use FKGL to measure readability; however, it does not take into account grammaticality and meaning preservation.

We compute SARI, BLEU, and FKGL using EASSE (Alva-Manchego et al., 2019)[5], a simplification evaluation library.

4.3 Training Details

We performed hyperparameters search using Optuna (Akiba et al., 2019) with T5-small and reduced

[5] https://github.com/feralvam/easse

size dataset to speed up the process. All models are trained with the same hyperparameters such as a batch size of 6 for T5-base and 12 for T5-small, maximum token of 256, learning rate of 3e-4, weight decay of 0.1, Adam epsilon of 1e-8, 5 warm up steps, 5 epochs, and the rest of the parameters are left with default values from Transformers library. Also, the seed is set to 12 for reproducibility. For the generation, we use beam size of 8. Our models are trained and evaluated using Google Colab Pro, which has a random GPU T4 or P100. Both have 16GB of memory, up to 25GB of RAM, and a time limit of 24h maximum for the execution of cells. Training of T5-base model for 5 epochs usually takes around 20 hours.

4.4 Choosing Control Token Values at Inference

In this experiment, we want to search for control token values that make the model generate the best possible simplifications. Thus, we select the values that achieve the best SARI on the validation set using the same tool that we use for hyperparameters tuning, Optuna (Akiba et al., 2019), and keep those values fixed for sentences in the test set. We repeat the same process for each evaluation dataset.

4.5 Baselines

We benchmark our model against several well-known state-of-the-art systems:

YATS (Ferrés et al., 2016)[6] Rule-based system with linguistically motivated rule-based syntactic analysis and corpus-based lexical simplifier which generates sentences based on part-of-speech tags and dependency information.

PBMT-R (Wubben et al., 2012) Phrase-based MT system trained on a monolingual parallel corpus with candidate re-ranking based on dissimilarity using Levenshtein distance.

UNTS (Surya et al., 2019) Unsupervised Neural Text Simplification is based on the encode-attend-decode style architecture (Bahdanau et al., 2014) with a shared encoder and two decoders and trained on unlabeled data extracted from English Wikipedia dump.

Dress-LS (Zhang and Lapata, 2017) A Seq2Seq model trained with deep reinforcement learning combined with a lexical simplification model to improve complex word substitutions.

DMASS+DCSS (Zhao et al., 2018b) A Seq2Seq model trained with the original Transformer architecture (Vaswani et al., 2017) combined with the simple paraphrase database for simplification PPDB. (Pavlick and Callison-Burch, 2016b).

ACCESS (Martin et al., 2020b) Seq2Seq system trained with four control tokens attached to source sentence: character length ratio, Levenshtein similarity ratio, word rank ratio, and dependency tree depth ratio between source and target sentence.

BART+ACCESS (Martin et al., 2020a) The system fine-tunes BART (Lewis et al., 2020) and adds the simplification control tokens from ACCESS.

4.6 Results

We evaluate our models automatically on two different datasets TurkCorpus and ASSET. In addition, we also perform a human evaluation on one of our models, which is described in Section 5. Table 2 reports the results of automatic evaluation of our models compared with other state-of-the-art systems. Our model **T5-base**+#chars+WordRank+LevSim+DepTreeDepth performs best on TurkCorpus with the SARI score of 43.31, while the other model **T5-base**+All Tokens performs best on ASSET with SARI score of 45.04 compared to the current state-of-the-art BART+ACCESS with the SARI score of 42.62 on TurkCorpus and 43.63 on ASSET. Following these results, our models out-perform all the state-of-the-art models in the literature in all approaches: rule-based, supervised and unsupervised approach even without using any additional resources.

5 Human Evaluation

In addition to automatic evaluation, we performed a human evaluation on the outputs of different systems. Following recent works (Alva-Manchego et al., 2017; Dong et al., 2019; Zhao et al., 2020), we run our evaluation on Amazon Mechanical Turk by asking five workers to rate using 5-point likert scale on three aspects: (1) Fluency (or Grammaticality): is it grammatically correct and well-formed?, (2) Simplicity: is it simpler than the original sentence?, and (3) Adequacy (or Meaning preservation): does it preserve meaning of the original sentence? More detailed instructions can be found in Appendix A. For this evaluation, we

[6] http://able2include.taln.upf.edu

Model	Data	ASSET			TurkCorpus		
		SARI↑	BLEU↑	FKGL↓	SARI↑	BLEU↑	FKGL↓
YATS	Rule-based	34.4	72.07	7.65	37.39	74.87	7.67
PBMT-R	PWKP (Wikipedia)	34.63	79.39	8.85	38.04	82.49	8.85
UNTS	Unsup. Data	35.19	76.14	7.60	36.29	76.44	7.60
Dress-LS	WikiLarge	36.59	86.39	7.66	36.97	81.08	7.66
DMASS+DCSS	WikiLarge	38.67	71.44	7.73	39.92	73.29	7.73
ACCESS	WikiLarge	40.13	75.99	7.29	41.38	76.36	7.29
BART+ACCESS	WikiLarge	43.63	76.28	6.25	42.62	78.28	6.98
T5-base+#Chars+WordRank							
+LevSim+DepTreeDepth	WikiLarge	44.91	71.96	6.32	**43.31**	66.23	6.17
T5-base+All Tokens	WikiLarge	**45.04**	71.21	5.88	43.00	64.42	5.63

Table 2: We report SARI, BLEU and FKGL evaluation results of our model compared with others on TurkCorpus and ASSET test set (SARI and BLEU higher the better, FKGL lower the better). BLEU and FKGL scores are not quite relevant for sentence simplification, and we keep them just to compare with the previous models. All the results of the literature are taken from Martin et al. (2020a), except YATS which is generated using its web interface.

randomly select 100 sentences from different simplification systems trained on WikiLarge dataset, except YATS which is rule-based. Table 3 reports the results in averaged values.

Model	Fluency	Simplicity	Adequacy
YATS	4.03*	3.62*	3.92*
DMASS+DCSS	3.84*	3.70*	3.48*
BART+ACCESS	**4.41**	**4.02**	4.13
Our Model	4.30	3.99	**4.18**

Table 3: Results of human evaluation on 100 random sentences selected from TurkCorpus test set. Best results are marked in bold, and results marked with an '*' are significantly lower than our model according to pared t-test with $p<0.01$. Our model in use here is **T5-base+All Tokens**.

The results have shown that our model performs lower in fluency and about the same in simplicity, and better in adequacy compared to BART+ACCESS. Based on our observation, there are two reasons that humans rated our model lower on fluency: (1) our model generates incorrect text format (without spaces) in some sentences (examples in Table 4). The problem can be easily spotted by human, but it does not affect the automatic evaluation as EASSE uses a tokenizer which can split the whole sentence correctly. (2) Our model tends to produce longer sentences than BART+ACCESS and in some cases, the subject is repeated twice when the sentence is split into two (e.g., relative clause). The repetition is also considered as one of the key features of simplification as it makes text easier to understand, but for native or fluent language speakers, repetition and the longer sentence make the fluency worse. Moreover, due to these problems, the evaluators also tend to lower the simplicity score as they consider it harder to read.

Sentence
So far the'celebrity'episodes have included Vic Reeves, Nancy Sorrell, and Gaby Roslin.
New South Wales'biggest city and capital is Sydney.

Table 4: Examples of incorrect text format generated by our model.

6 Ablation Study

In this section, we investigate the contribution of each token and different T5 pre-trained models to the performance of the system. Table 5 reports the scores of models trained on WikiLarge and evaluated with TurkCorpus and ASSET test set. Table 6 shows all control token values used for all

Model	ASSET			TurkCorpus		
	SARI↑	BLEU↑	FKGL↓	SARI↑	BLEU↑	FKGL↓
T5-small (No tokens)	29.85	90.39	8.94	34.50	94.16	9.44
T5-small + All Tokens	39.12	86.08	6.99	40.83	85.12	6.78
T5-base (No tokens)	34.15	88.97	8.94	37.56	90.96	8.81
T5-base:						
+#Words	38.51	84.02	7.45	38.86	89.10	8.61
+#Chars	39.58	79.22	6.06	38.95	84.81	7.76
+LevSim	41.58	82.52	6.53	40.90	85.45	7.55
+WordRank	41.40	76.75	5.85	41.44	85.46	7.67
+DepTreeDepth	40.08	81.94	6.56	39.18	87.60	7.81
T5-base:						
+WordRank+LevSim	42.85	80.38	4.47	41.75	83.90	7.42
+#Chars+WordRank+LevSim	44.89	56.76	5.93	42.91	67.09	6.53
+#Words+#Chars+WordRank+LevSim	44.65	58.52	5.52	43.03	68.11	5.96
+#Chars+WordRank+LevSim+DepTreeDepth	44.91	71.96	6.32	**43.31**	66.23	6.17
+All Tokens	**45.04**	71.21	5.88	43.00	64.42	5.63

Table 5: Ablation study on different T5 models and different control token values. Each model is trained and evaluated independently. We report SARI, BLEU and FKGL on TurkCorpus and ASSET test set. Control token values corresponded to each model are listed in the Table 6

Model	ASSET	TurkCorpus
T5-small (No tokens)		
T5-small + All Tokens	$W_{1.05}$ $C_{0.95}$ $WR_{0.75}$ $L_{0.75}$ $DTD_{0.75}$	$W_{1.05}$ $C_{0.95}$ $WR_{0.85}$ $L_{0.85}$ $DTD_{0.85}$
T5-base (No tokens)		
T5-base:		
+#Words	$W_{0.75}$	$W_{0.85}$
+#Chars	$C_{0.5}$	$C_{0.75}$
+LevSim	$L_{0.75}$	$L_{0.85}$
+WordRank	$WR_{0.25}$	$WR_{0.85}$
+DepTreeDepth	$DTD_{0.5}$	$DTD_{0.75}$
T5-base:		
+WordRank+LevSim	$W_{0.75}$ $L_{0.75}$	$W_{0.85}$ $L_{0.85}$
+#Chars+WordRank+LevSim	$C_{0.95}$ $WR_{0.75}$ $LevSim_{0.75}$	$C_{0.95}$ $WR_{0.85}$ $L_{0.85}$
+#Words+#Chars+WordRank+LevSim	$W_{1.05}$ $C_{0.95}$ $WR_{0.75}$ $L_{0.75}$	$W_{1.05}$ $C_{0.95}$ $WR_{0.75}$ $L_{0.75}$
+#Chars+WordRank+LevSim+DepTreeDepth	$C_{0.95}$ $WR_{0.75}$ $L_{0.75}$ $DTD_{0.75}$	$C_{0.95}$ $WR_{0.75}$ $L_{0.75}$ $DTD_{0.75}$
+All Tokens	$W_{1.05}$ $C_{0.95}$ $WR_{0.75}$ $L_{0.75}$ $DTD_{0.75}$	$W_{1.05}$ $C_{0.95}$ $WR_{0.85}$ $L_{0.85}$ $DTD_{0.85}$

Table 6: These are the control token values used for the ablation study in Table 5. Each model is trained and evaluated independently. The values are selected using the hyperparameters search tool mentioned in Section 4.4.

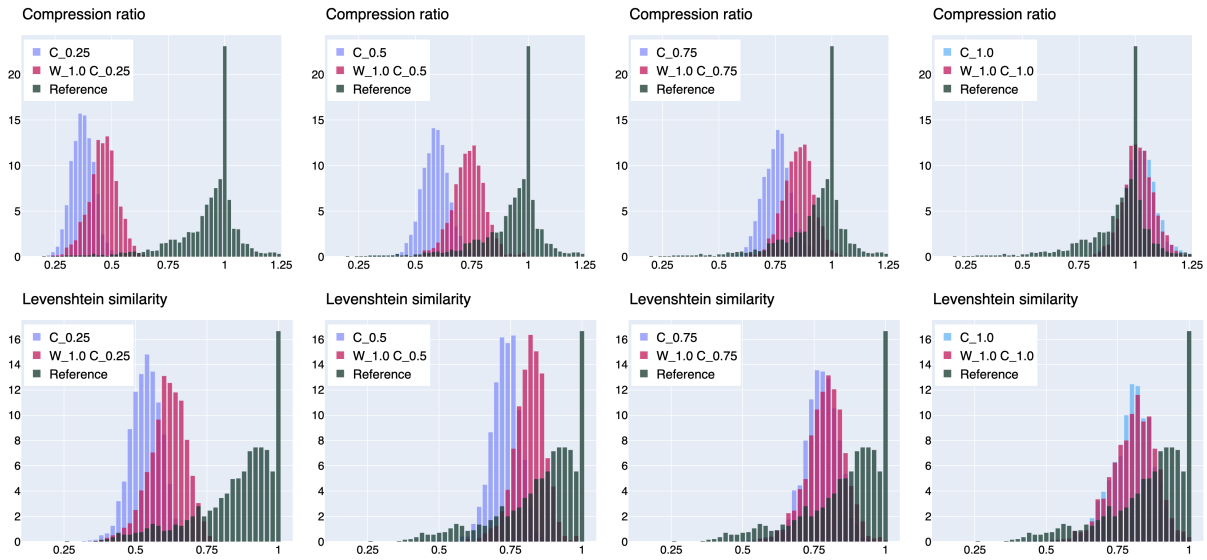


Figure 1: Influence of #Words and #Chars control tokens on the simplification outputs. Red represents the outputs of the model trained with four tokens, without #Words control token. Blue represents the outputs of the model trained with all five tokens. Green is the reference taken from TurkCorpus. The first row shows the compression ratio (number of chars ratio between system outputs and source sentences), and second row is the Levenshtein similarity (words similarity between system outputs and source sentences) of each model. We plot the results of the 2000 validation sentences from TurkCorpus. Other control token values used here are set to 0.75, the example in Table 7.

the models in Table 5 which are selected using the same process and tool as mentioned in Section 4.4.

Based on the results, the larger model (T5-base) performs better than the smaller one (T5-small) on both datasets (+3.06 on TurkCorpus, +4.3 on ASSET). It is due to the fact that larger model has more information which could generate better and more coherent text. Moreover, when added control tokens, the performance increases significantly. With only one token, WordRank performs best on TurkCorpus (+3.88 over T5-base) and LevSim on ASSET (+7.43 over T5-base).

Using pre-trained model alone does not gain much improvement, only when combined with control tokens, the results improve by a big margin (+3.06 and +9.28 for T5-small with and without tokens), and (+5.75 and +10.89 for T5-base with and without tokens).

6.1 Analysis on the effect of #Words

Our goal of using #Words control token is to make the model learn to generate shorter words whereas #Chars alone could help the model regulate the sentence length but not word length, so here we investigate how #Words and #Chars control tokens affect the outputs.

For the model with #Words token to work, it has to be incorporated with #Chars as #Words determines the number of words and #Chars limits the number of characters in the sentence. In our examples Table 7, we set #Words to 1.0, which means the number of words in the simplified sentence has to be similar to the original sentence, and #Chars is set to 0.5 and 0.75, which means keeping the same amount of words but reduces 50% or 25% of characters.

Figure 1 shows the differences in density distribution (first row) and similarity (second row) between model 1 in red without #Words token, model 2 in blue with #Words tokens, and the one in green is the reference. The first column #Chars is set to 0.25, second column #Chars=0.5, third column #Chars=0.75, fourth #Chars=1.0, and in all cases #words is set to 1.0. From the plots, we can see that model 1 does more compression than model 2, which means model 2 preserve more words than model 1.

Table 7 shows some example sentences comparing models with #Chars_0.75 and #Chars_0.5. When #Chars is set to 0.75, we do not see much difference between the two models, but when #Chars is set to 0.5, the two models have differences in terms of sentence length and word length. For example, the word **mathematics** in the example number one is replaced with the word **math** in model 2 (with #Words) and removed by model 1

Tokens	Model 1: **#Chars_0.5** WordRank_0.75 LevSim_0.75 DepTreeDepth_0.75 Model 2: **#Words_1.0 #Chars_0.5** WordRank_0.75 LevSim_0.75 DepTreeDepth_0.75
Source:	In order to accomplish their objective, surveyors use elements of geometry, engineering, trigonometry, **mathematics**, physics, and law.
Model 1:	In order to accomplish their objective, surveyors use geometry, engineering, and law.
Model 2:	In order to do this, surveyors use geometry, engineering, trigonometry, **math**, physics, and law.
Source:	The **municipality** has about 5700 inhabitants.
Model 1:	The municipality has 5700.
Model 2:	The **town** has about 5700.
Source:	A hunting dog refers to any dog who **assists** humans in hunting.
Model 1:	A hunting dog is any dog who hunts.
Model 2:	A hunting dog is a dog who **helps** humans in hunting.
Tokens	Model 1: **#Chars_0.75** WordRank_0.75 LevSim_0.75 DepTreeDepth_0.75 Model 2: **#Words_1.0 #Chars_0.75** WordRank_0.75 LevSim_0.75 DepTreeDepth_0.75
Source:	The park has become a traditional **location** for **mass demonstrations**.
Model 1:	The park has become a popular **place** for **demonstrations**.
Model 2:	The park has become a **place** for **people to show things**.
Source:	Frances was later **absorbed** by an **extratropical** cyclone on November 21.
Model 1:	Frances was later **taken** in by an extratropical cyclone.
Model 2:	Frances was later **taken** over by a cyclone on November 21.
Source:	There are claims that thousands of people were **impaled** at a single time.
Model 1:	There are claims that thousands of people were **killed**.
Model 2:	There are also stories that thousands of people were **killed** at a time.

Table 7: Examples showing the differences between the model with number of words ratio versus the one without. Model 1 trained with four tokens, without #Words control token, and model 2 trained with all five control tokens. All control token values used to generate the outputs are listed in the rows Tokens. We use bold to highlight the differences.

(without #Words). Second example, the word **municipality** is replaced by the word **town** by model 2, and model 1 simply keeps the word and crops the sentence (the same problem with the third example). In addition, the fourth example, the word **location** is replaced by both models with the word **place**, the phrase **mass demonstration** is reduced to **demonstration** by the model 1 whereas model 2 changes to four shorter words **people to show things**.

There are many cases where model 1 and model 2 generate the same substitutions, but very often model 1 tends to crop the end of the sentence or drops some words to fulfill the length constraint. Whereas model 2 tends to generate longer sentences than model 1, less crop, and very often replaces long complex words with shorter ones. Even though, based on the results from Table 2, adding the #Words control token does not significantly improve the SARI score and sometimes even lowers the score, it certainly holds its purpose.

7 Conclusion

In this paper, we propose a method which leverages a big pre-trained model (T5) fine-tuning it for the Controllable Sentence Simplification task. The experiments have shown good results of 43.31 SARI on TurkCorpus evaluation set and of 45.04 on ASSET evaluation set, outperforming the current state-of-the-art model. Also, we have shown that adding the control token #Words is useful for generating substitutions with a shorter lengths.

Acknowledgments

We acknowledge support from the project Context-aware Multilingual Text Simplifi-

cation (ConMuTeS) PID2019-109066GB-I00/AEI/10.13039/501100011033 awarded by Ministerio de Ciencia, Innovación y Universidades (MCIU) and by Agencia Estatal de Investigación (AEI) of Spain. Also, we would like to thank the three anonymous reviewers for their insightful suggestions.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

Fernando Alva-Manchego, Joachim Bingel, Gustavo Paetzold, Carolina Scarton, and Lucia Specia. 2017. Learning how to simplify from explicit labeling of complex-simplified text pairs. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 295–305.

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 2020. ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4668–4679, Online. Association for Computational Linguistics.

Fernando Alva-Manchego, Louis Martin, Carolina Scarton, and Lucia Specia. 2019. EASSE: Easier automatic sentence simplification evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 49–54, Hong Kong, China. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Eduard Barbu, M Teresa Martín-Valdivia, Eugenio Martínez-Cámara, and L Alfonso Ureña-López. 2015. Language technologies applied to document simplification for helping autistic people. *Expert Systems with Applications*, 42(12):5076–5086.

Delphine Bernhard, Louis De Viron, Véronique Moriceau, and Xavier Tannier. 2012. Question generation for french: collating parsers and paraphrasing questions. *Dialogue & Discourse*, 3(2):43–74.

John A Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*.

Raman Chandrasekar, Christine Doran, and Srinivas Bangalore. 1996. Motivations and methods for text simplification. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

William Coster and David Kauchak. 2011. Simple english wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–669.

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

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2019. Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing. *arXiv preprint arXiv:1906.08104*.

Biljana Drndarević and Horacio Saggion. 2012. Towards automatic lexical simplification in spanish: an empirical study. In *Proceedings of the First Workshop on Predicting and Improving Text Readability for target reader populations*, pages 8–16.

Richard J Evans. 2011. Comparing methods for the syntactic simplification of sentences in information extraction. *Literary and linguistic computing*, 26(4):371–388.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *arXiv preprint arXiv:1711.05217*.

Daniel Ferrés, Montserrat Marimon, Horacio Saggion, et al. 2016. Yats: yet another text simplifier. In *International Conference on Applications of Natural Language to Information Systems*, pages 335–342. Springer.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*.

Siddhartha Jonnalagadda and Graciela Gonzalez. 2010. Biosimplify: an open source sentence simplification engine to improve recall in automatic biomedical information extraction. In *AMIA Annual Symposium Proceedings*, volume 2010, page 351. American Medical Informatics Association.

Oleg Kariuk and Dima Karamshuk. 2020. Cut: Controllable unsupervised text simplification. *arXiv preprint arXiv:2012.01936*.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 1537–1546.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. *arXiv preprint arXiv:1609.09552*.

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

Reno Kriz, Joao Sedoc, Marianna Apidianaki, Carolina Zheng, Gaurav Kumar, Eleni Miltsakaki, and Chris Callison-Burch. 2019. Complexity-weighted loss and diverse reranking for sentence simplification. *arXiv preprint arXiv:1904.02767*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Louis Martin, Angela Fan, Éric de la Clergerie, Antoine Bordes, and Benoît Sagot. 2020a. Multilingual unsupervised sentence simplification. *arXiv preprint arXiv:2005.00352*.

Louis Martin, Éric Villemonte de La Clergerie, Benoît Sagot, and Antoine Bordes. 2020b. Controllable Sentence Simplification. In *LREC 2020 - 12th Language Resources and Evaluation Conference*, Marseille, France. Due to COVID19 pandemic, the 12th edition is cancelled. The LREC 2020 Proceedings are available at http://www.lrec-conf.org/proceedings/lrec2020/index.html.

Kerstin Matausch and Birgit Peböck. 2010. Easyweb–a study how people with specific learning difficulties can be supported on using the internet. In *International Conference on Computers for Handicapped Persons*, pages 641–648. Springer.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 85–91.

Gustavo H Paetzold and Lucia Specia. 2016. Unsupervised lexical simplification for non-native speakers. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3761–3767.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ellie Pavlick and Chris Callison-Burch. 2016a. Simple ppdb: A paraphrase database for simplification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–148.

Ellie Pavlick and Chris Callison-Burch. 2016b. Simple PPDB: A paraphrase database for simplification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–148, Berlin, Germany. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Luz Rello, Ricardo Baeza-Yates, Stefan Bott, and Horacio Saggion. 2013a. Simplify or help? text simplification strategies for people with dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, pages 1–10.

Luz Rello, Susana Bautista, Ricardo Baeza-Yates, Pablo Gervás, Raquel Hervás, and Horacio Saggion. 2013b. One half or 50%? an eye-tracking study of number representation readability. In *IFIP Conference on Human-Computer Interaction*, pages 229–245. Springer.

Horacio Saggion. 2017. Automatic Text Simplification. *Synthesis Lectures on Human Language Technologies*, 10(1):1–137.

Carolina Scarton and Lucia Specia. 2018. Learning simplifications for specific target audiences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 712–718.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of*

the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 35–40.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 896–902, Geneva, Switzerland. COLING.

Sanja Štajner, Iacer Calixto, and Horacio Saggion. 2015. Automatic text simplification for spanish: Comparative evaluation of various simplification strategies. In *Proceedings of the international conference recent advances in natural language processing*, pages 618–626.

Sanja Štajner and Maja Popović. 2016. Can text simplification help machine translation? In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 230–242.

Sanja Štajner and Maja Popović. 2019. Automated text simplification as a preprocessing step for machine translation into an under-resourced language. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1141–1150.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018. BLEU is not suitable for the evaluation of text simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 738–744, Brussels, Belgium. Association for Computational Linguistics.

Sai Surya, Abhijit Mishra, Anirban Laha, Parag Jain, and Karthik Sankaranarayanan. 2019. Unsupervised neural text simplification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2058–2068, Florence, Italy. Association for Computational Linguistics.

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

Tu Vu, Baotian Hu, Tsendsuren Munkhdalai, and Hong Yu. 2018. Sentence simplification with memory-augmented neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 79–85, New Orleans, Louisiana. Association for Computational Linguistics.

Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: reading assistance for low-literacy readers. In *Proceedings of the 27th ACM international conference on Design of communication*, pages 29–36.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 409–420.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1015–1024, Jeju Island, Korea. Association for Computational Linguistics.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594, Copenhagen, Denmark. Association for Computational Linguistics.

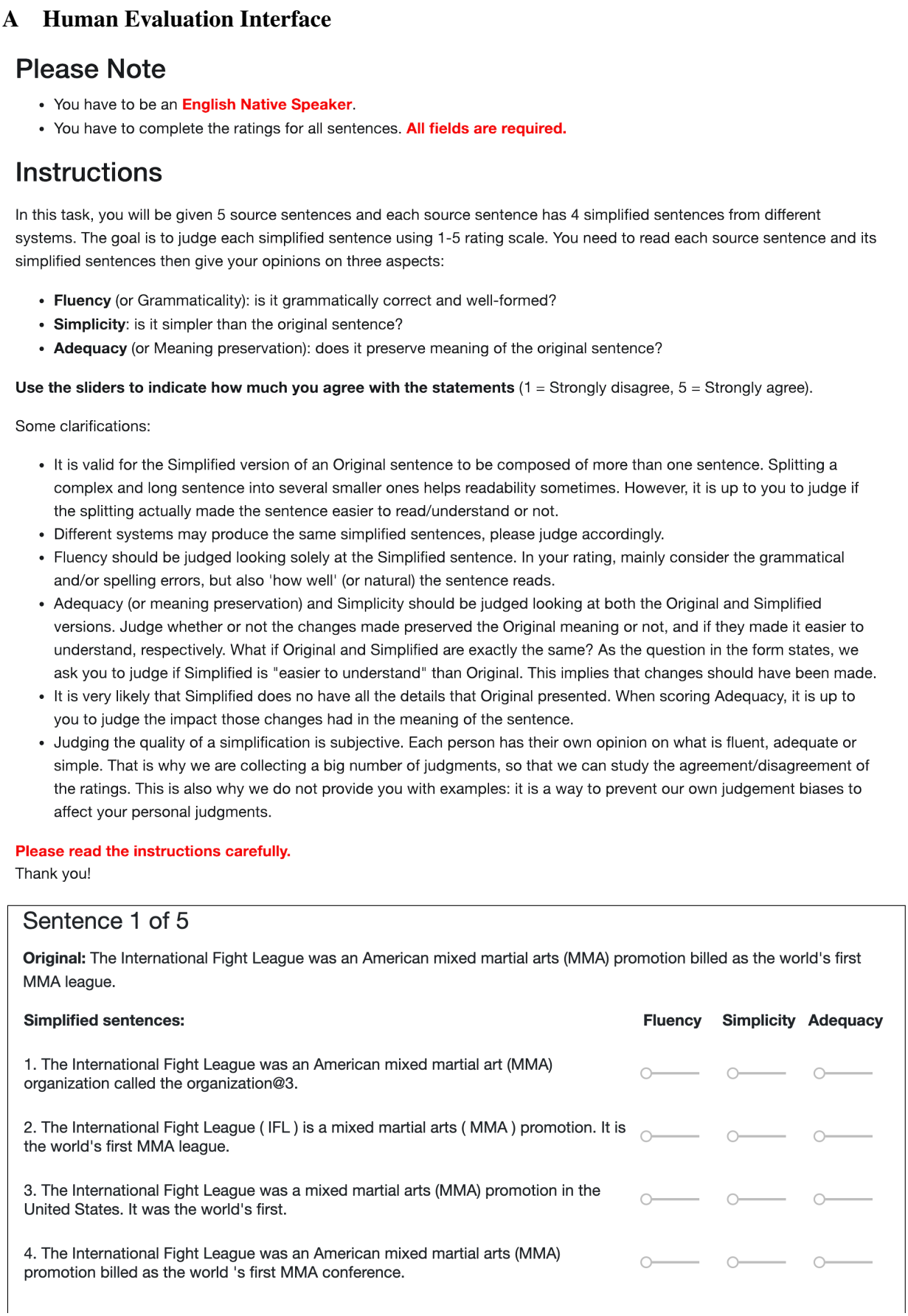
Sanqiang Zhao, Rui Meng, Daqing He, Saptono Andi, and Parmanto Bambang. 2018a. Integrating transformer and paraphrase rules for sentence simplification. *arXiv preprint arXiv:1810.11193*.

Sanqiang Zhao, Rui Meng, Daqing He, Andi Saptono, and Bambang Parmanto. 2018b. Integrating transformer and paraphrase rules for sentence simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3164–3173, Brussels, Belgium. Association for Computational Linguistics.

Yanbin Zhao, Lu Chen, Zhi Chen, and Kai Yu. 2020. Semi-supervised text simplification with back-translation and asymmetric denoising autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9668–9675.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361.

A Human Evaluation Interface



Please Note

- You have to be an **English Native Speaker**.
- You have to complete the ratings for all sentences. **All fields are required.**

Instructions

In this task, you will be given 5 source sentences and each source sentence has 4 simplified sentences from different systems. The goal is to judge each simplified sentence using 1-5 rating scale. You need to read each source sentence and its simplified sentences then give your opinions on three aspects:

- **Fluency** (or Grammaticality): is it grammatically correct and well-formed?
- **Simplicity**: is it simpler than the original sentence?
- **Adequacy** (or Meaning preservation): does it preserve meaning of the original sentence?

Use the sliders to indicate how much you agree with the statements (1 = Strongly disagree, 5 = Strongly agree).

Some clarifications:

- It is valid for the Simplified version of an Original sentence to be composed of more than one sentence. Splitting a complex and long sentence into several smaller ones helps readability sometimes. However, it is up to you to judge if the splitting actually made the sentence easier to read/understand or not.
- Different systems may produce the same simplified sentences, please judge accordingly.
- Fluency should be judged looking solely at the Simplified sentence. In your rating, mainly consider the grammatical and/or spelling errors, but also 'how well' (or natural) the sentence reads.
- Adequacy (or meaning preservation) and Simplicity should be judged looking at both the Original and Simplified versions. Judge whether or not the changes made preserved the Original meaning or not, and if they made it easier to understand, respectively. What if Original and Simplified are exactly the same? As the question in the form states, we ask you to judge if Simplified is "easier to understand" than Original. This implies that changes should have been made.
- It is very likely that Simplified does no have all the details that Original presented. When scoring Adequacy, it is up to you to judge the impact those changes had in the meaning of the sentence.
- Judging the quality of a simplification is subjective. Each person has their own opinion on what is fluent, adequate or simple. That is why we are collecting a big number of judgments, so that we can study the agreement/disagreement of the ratings. This is also why we do not provide you with examples: it is a way to prevent our own judgement biases to affect your personal judgments.

Please read the instructions carefully.
Thank you!

Sentence 1 of 5

Original: The International Fight League was an American mixed martial arts (MMA) promotion billed as the world's first MMA league.

Simplified sentences:	**Fluency**	**Simplicity**	**Adequacy**
1. The International Fight League was an American mixed martial art (MMA) organization called the organization@3.			
2. The International Fight League (IFL) is a mixed martial arts (MMA) promotion. It is the world's first MMA league.			
3. The International Fight League was a mixed martial arts (MMA) promotion in the United States. It was the world's first.			
4. The International Fight League was an American mixed martial arts (MMA) promotion billed as the world 's first MMA conference.			

Figure 2: Our interface is based on the one proposed by Kriz et al. (2019), and the consent form based on Alva-Manchego et al. (2020).

SEPRG: Sentiment aware Emotion controlled Personalized Response Generation

Mauajama Firdaus, Umang Jain, Asif Ekbal and Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology Patna, India
(mauajama.pcs16,umang,asif,pb)@iitp.ac.in

Abstract

Social chatbots have gained immense popularity, and their appeal lies not just in their capacity to respond to the diverse requests from users, but also in the ability to develop an emotional connection with users. To further develop and promote social chatbots, we need to concentrate on increasing user interaction and take into account both the intellectual and emotional quotient in the conversational agents. Therefore, in this work, we propose the task of sentiment aware emotion controlled personalized dialogue generation giving the machine the capability to respond emotionally and in accordance with the persona of the user. As sentiment and emotions are highly co-related, we use the sentiment knowledge of the previous utterance to generate the correct emotional response in accordance with the user persona. We design a Transformer based Dialogue Generation framework, that generates responses that are sensitive to the emotion of the user and corresponds to the persona and sentiment as well. Moreover, the persona information is encoded by a different Transformer encoder, along with the dialogue history, is fed to the decoder for generating responses. We annotate the PersonaChat dataset with sentiment information to improve the response quality. Experimental results on the PersonaChat dataset show that the proposed framework significantly outperforms the existing baselines, thereby generating personalized emotional responses in accordance with the sentiment that provides better emotional connection and user satisfaction as desired in a social chatbot.

1 Introduction

One of the significant challenges of artificial intelligence (AI) is to endow the machine with the ability to interact in natural language with humans. In the recent past, tremendous effort has been given to create smart personal assistants, such as Microsoft's Cortana, Apple's Siri, Amazon's Alexa, Google Home, etc. The personal assistants in our mobile devices invariably assist in our day-to-day lives by answering a wide range of queries. Such assistants act as social agents that take care of the various activities of their users. Besides reacting passively to user requests, they also proactively anticipate user needs and provide in-time assistance, such as reminding of an upcoming event or suggesting a useful service without receiving explicit user requests (Sarikaya, 2017). The daunting task for these agents is that they have to work well in open domain scenarios as people learn to rely on them to effectively maintain their works and lives efficiently.

Empathy and social belonging are a few of the fundamental needs for human beings (Maslow, 1943). Building social chatbots to tackle these emotional needs is indeed of great benefit to our society. The primary objective of these chatbots is not inherently to answer all the users' questions, but rather to be a virtual companion of the users. To become a better companion, it is imperative for the agent to understand the personality of the user to assist in different aspects of life. Along with understanding the personality traits of a user, the emotional connection is an essential factor for building better communication. Social conversational agents can serve for a more extended period of time by maintaining a consistent personality (increasing trust in the user) and by establishing an emotional connection with them. The ability to empathize, create social belonging, and adhere to a personality and integration of these factors in conversational agents is one of the long-standing goals of Artificial Intelligence (AI). Conversational agents need to monitor the user's emotion in order to suffice the emotional needs and simultaneously empathize with them, making the conversation engaging, increasing user contentment (Prendinger et al., 2005),

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 353–363,
Aberdeen, Scotland, UK, 20-24 September 2021.

Persona 1	Persona 2
I am primarily a meat eater.	*I've a sweet tooth.*
I am a guitar player.	*I'm a babysitter and drive a mercedes.*
Welding is my career field.	*I'm the middle child of 3 siblings.*
My parents don't know I am gay.	*I'm getting married in six months.*
[Person 1] What do you do for career? (Neutral)	
[Person 2] I like to watch kids. (Positive)	
[Person 1] I actually play guitar and do a lot of welding. (Positive)	
[Person 2] What do you weld? houses?(Neutral)	

Table 1: A conversation from the PersonaChat dataset with sentiments

and decreasing breakdowns in conversations (Martinovski and Traum, 2003). Moreover, these agents should also have the capability to generate personalized responses conforming to the personal interests and unique needs of different users while presenting a consistent personality to gain the user's trust and confidence. Hence, the primary motivation of our current work lies in generating responses that are engaging, emotionally appropriate, and also integrates the personal interests of the user.

Lately, researchers have started focusing on incorporating personality information on chit-chat (Zhang et al., 2018) and goal-oriented (Joshi et al., 2017; Luo et al., 2019) conversational systems. Due to the lack of persona data sets, the authors created a PersonaChat dataset in (Zhang et al., 2018), where the individual personality data is represented in a few texts for open-domain chit-chat dialogue systems. We present an example from the dataset in Table 1, from which it is obvious that the speakers are able to retain the persona knowledge when communicating with each other. This helps to make the dialogue engaging and also makes it easier to build trust and credibility with the users (Shum et al., 2018). For conversational systems to effectively communicate with the user in a coherent and natural way, the ability to maintain a clear persona is imperative. While it is necessary to maintain a clear personality in order to gain the confidence of the user, it is also essential to react emotionally in order to create a bond with the user.

From Table 1, it is evident that when talking with the user, the agent can retain a specific personality, but it sacrifices the emotional link with the user. The dialogue, therefore, is almost like stating facts instead of a real discussion. In this work, therefore, we propose the task of infusing the responses with emotional content while maintaining a clear persona. From the table, the response to *Person 2* could be more empathetic like *That's a great job, as I play guitar and do welding for a career.* This response has a happy undertone than the ground-truth response, which is neutral and contains only the facts about *Person 1*. Empathetic responses are insightful and provide a forum for a more substantial discussion. It is evident from the illustration that only having a persona in a reply is not sufficient to produce interactive responses. To render it more human-like, the emotional element must also be integrated into the replies. Emotions and sentiments are subjective qualities and are understood to share overlapping features; hence are frequently used interchangeably.

This is mainly because both sentiment and emotion refer to experiences resulting from the combination of biological, cognitive, and social influences. Though both are considered to be the same, yet according to (Munezero et al., 2014), the sentiment is formed and retained for a longer duration, whereas emotions are like episodes that are shorter in length. Moreover, the sentiment is mostly target-centric, while emotions are not always directed to a target. Every emotion is associated with sentiments, hence using the sentiment information of the utterances can assist in narrowing down the set of emotions for generating contextually correct emotional responses. In the Table 1, the dialogue has been annotated with the corresponding sentiments to assist in generating empathetic responses. To the best of our knowledge, this is one of the first works that include sentiment information for creating personalized emotional responses.

The key contributions of this work are as follows:

1. We propose the task of generating empathetic, personalized responses while considering the persona information and implicitly the sentiment in the responses through the dialogue context.

2. We propose a novel Transformer based encoder-decoder framework, with the ability to infuse the sentiment, emotion and persona information in the responses.

3. Experimental results show that our proposed framework is capable of maintaining a consistent persona and sentiment while generating emotional responses compared to the existing baselines.

The rest of the paper is structured as follows. In Section II, we present a brief survey of the related work. In Section III, we explain the proposed

methodology. In Section IV, we describe the details of the datasets that we used and annotated. The experimental setup, along with the evaluation metrics, is reported in Section V. In Section VI, we present the results along with the necessary analysis. Finally, we conclude in Section VII with future work.

2 Related Work

In complete applications, such as dialogue systems, natural language generation (NLG) has become increasingly essential (Vinyals and Le, 2015; Li et al., 2016b; Serban et al., 2017; Wu et al., 2018) and also in many other natural language interfaces. The generation of responses provides the means by which a conversational agent can communicate with its user to assist users in achieving their desired goals. Recently, generative adversarial networks have been exploited for dialogue generation (Xu et al., 2018, 2017; Zhang et al., 2019; Zhu et al., 2019; Bruni and Fernandez, 2017) for a better generation of responses.

Persona information is an essential part of generating responses. Earlier works on persona-based conversational models (Li et al., 2016a) incorporated speakers' embeddings to infuse persona information in the responses. To incorporate persona in chit-chat models, the authors in (Zhang et al., 2018; Mazaré et al., 2018) introduced a PersonaChat dataset that includes personal information of the speakers. This dataset has been extensively used to build persona-based dialogue systems (Madotto et al., 2019; Yavuz et al., 2019; Song et al., 2019, 2020). The authors in (Madotto et al., 2019) used a meta-learning framework to include persona information in the generated responses. Similarly, the authors in (Yavuz et al., 2019) employed a hierarchical pointer network for generating persona-based responses. The authors in (Song et al., 2019) used persona information to generate diverse responses by employing conditional variational autoencoder. Our present work differs from these existing works (that made use of the PersonaChat dataset) in a sense that we intend to use the persona information while generating emotional responses.

Persona information is also being exploited in goal-oriented dialogue systems (Joshi et al., 2017; Luo et al., 2019; Qian et al., 2017). The authors in (Joshi et al., 2017) introduced persona information in the babI dialog dataset for creating better responses. The authors used conditional variational auto-encoders for personalized generation in (Wu et al., 2020). As personalization has been considered in responses, we intend to take a step ahead by inculcating the emotions in accordance to the emotion of the user and the dialogue history.

Lately, emotional text generation has gained immense popularity (Huang et al., 2018; Li and Sun, 2018; Lin et al., 2019; Li et al., 2017; Ghosh et al., 2017; Kezar, 2018; Rashkin et al., 2019; Zhou and Wang, 2017). In (Zhou et al., 2018), an emotional chatting machine (ECM) was proposed that was built upon seq2seq framework for generating emotional responses. Recently, a lexicon-based attention framework was employed to generate responses with a specific emotion (Song et al., 2020). Emotional embedding, along with affective sampling and regularizer, was employed to generate the affect driven dialogues in (Colombo et al., 2019). Lately, authors in (Firdaus et al., 2020) designed personalized response generation framework with controllable emotions using basic sequence-to-sequence framework. Our present research differs from these existing works as we propose a novel framework using a generative adversarial network to generate responses in an empathetic manner, having a consistent persona.

3 Methodology

We define the problem statement in this section, followed by the detailed descriptions of the proposed methodology. The architectural diagram of the sentiment and persona guided emotional dialogue generation framework is presented in Figure 1.

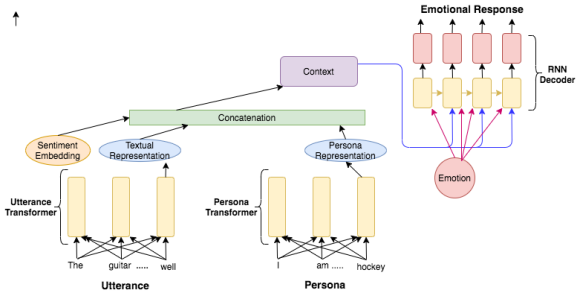


Figure 1: Architectural diagram of the proposed framework.

Problem Definition: In our present work we aim at solving the task of emotional and personalized dialogue generation in accordance to the conversational history, sentiment and the persona information of the speaker. For a given sequence

of dialogue turns $D = [U_1, U_2, \ldots, U_N]$ as the dialogue context, where $U_n = [w_1, w_2, \ldots, w_k]$ is the n^{th} dialogue turn and each dialogue turn is associated with sentiment labels represented by $S_{lab} = s_1, s_2, \ldots, s_N$ having a set of persona information $P = P_1, P_2, \ldots, P_m$ the task is to generate an emotional personalized response $Y = y_1, y_2, \ldots, y_{n'}$ along with the emotion embedding V_e for the desired emotion E that is sensitive to the speaker's expressed sentiment and is consistent to the persona information.

3.1 Proposed Framework

Our proposed framework is based upon the Transformer network (Vaswani et al., 2017) as shown in Figure 1. Our network comprises of two encoders: an utterance encoder to transform the textual utterance $U_i = (w_{k,1}, w_{k,2}, ..., w_{k,n})$ and a persona encoder to encode the set of persona information $P = P_1, P_2, \ldots, P_m$. Finally, we employ a transformer decoder to generate emotionally controlled responses according to the specified emotions in a similar manner as (Firdaus et al., 2020; Huang et al., 2018; Zhou et al., 2018).

Utterance Encoder: As the transformer encoder has multiple layers and each layer is composed of a multi-head self attentive sub-layer followed by a feed-forward sub-layer with residual connections (He et al., 2016) and layer normalization (Ba et al., 2016), we use it to encode the utterances in a given dialog. For intricate details on the Transformer network, we refer the interested readers to (Vaswani et al., 2017). To learn the representation of $U_i = (w_{k,1}, w_{k,2}, ..., w_{k,n})$ is first mapped into continuous space

$$T_u = (t_1^i, t_2^i, \ldots, t_{|U_i|}^i); where[T_j^i = e(w_j^i) + p_j] \quad (1)$$

where $e(u_j^i)$ and p_j are the word and positional embedding of every word u_j^i in an utterance, respectively. For words we use Glove embeddings and we adopt sine-cosine positional embedding (Vaswani et al., 2017) as it performs better and does not introduce additional trainable parameters. The utterance encoder (a Transformer) converts T_u into a list of hidden representations $h_1^i, h_2^i, \ldots, h_{|U_i|}^i$. We use the last hidden representation $h_{|U_i|}^i$ (i.e. the representation at the EOS token) as the textual representation of the utterance U_i. Similarly, to the representation of each word in U_i, we also take into account the utterance position. Therefore, the final textual representation of the utterance U_i is:

$$h_i^s = h_{|U_i|}^i + p_i \quad (2)$$

Note that the words and sentences share the same positional embedding matrix. We also concatenate the sentiment information of every sentences represented by $S_{lab} = s_1, s_2, \ldots, s_N$. The final representation of any utterance is given by the concatenation of the sentiment representation as well as the last hidden representation of the utterance.

$$h_i^{utt} = h_i^s + s_N \quad (3)$$

Persona Encoder: To learn the representation of the set of persona information $P = P_1, P_2, \ldots, P_m$ is first mapped into continuous space

$$T_p = (t_1^i, t_2^i, \ldots, t_{|P_m|}^i); where[T_k^i = e(w_k^i) + p'_k] \quad (4)$$

where $e(u_k^i)$ and p'_k are the word and positional embedding of every word u_k^i in a given persona, respectively. Similar to the utterance encoder, for words we use the Glove embeddings and adopt sine-cosine positional embedding (Vaswani et al., 2017). The persona encoder (a Transformer) converts T_p into a list of hidden representations ${h'_1}^i, {h'_2}^i, \ldots, {h'_{|P_m|}}^i$. We use the last hidden representation $h_{|P_m|}^i$ (i.e. the representation at the EOS token) as the persona representation of the given speaker. Therefore, the final persona representation of the utterance P_m is:

$$h_i^p = h_{|P_m|}^i + p'_i \quad (5)$$

Emotion controlled Decoder: To generate the next textual response with the given emotion information we employ a RNN decoder as shown in Figure 1. We employ GRU for generating the response in a sequential manner based on the context hidden representation from both the transformers, and the words decoded previously. We use the input feeding decoding along with the attention (Luong et al., 2015) mechanism for enhancing the performance of the model. Using the decoder state $h_{d,t}^{dec}$ as the query vector, we apply self-attention on the hidden representation of the utterance-level encoder. The decoder state, persona information and the context vector are concatenated and used to calculate a final distribution of the probability

over the output tokens.

$$
\begin{aligned}
h^{dec}_{d,t} &= GRU_d(y_{k,t-1}, h_{d,t-1}) \\
c_t &= \sum_{i=1}^{k} \alpha_{t,i} \hat{D}, \\
\alpha_{t,i} &= softmax(\hat{D}^T W_f h_{d,t}) \\
\tilde{h}_t &= tanh(W_{\tilde{h}}[h_{d,t}; c_t]) \\
P(y_t/y_{<t}) &= softmax(W_V \tilde{h}_t)
\end{aligned} \tag{6}
$$

where, W_f, W_V and $W_{\tilde{h}}$ are the trainable weight matrices.

For generating responses with the specified emotion as shown in Figure 1, we provide the emotion vector V_e (the emotion embeddings are pre-trained Glove embeddings) as input during decoding at every decoder time-step. In order to include the emotion vector in the decoder, we modify Equation (6) to incorporate the emotion information for the generation of responses and the modified equation is as follows:

$$
h^{dec}_{d,t} = GRU_d(y_{k,t-1}, [h_{d,t-1}, V_e]) \tag{7}
$$

Training and Inference: We employ commonly used teacher forcing (Williams and Zipser, 1989) algorithm at every decoding step to minimize the negative log-likelihood on the model distribution. We define $y^* = \{y^*_1, y^*_2, \ldots, y^*_m\}$ as the ground-truth output sequence for a given input by:

$$
\mathcal{L}_{ml} = -\sum_{t=1}^{m} \log p(y^*_t | y^*_1, \ldots, y^*_{t-1}) \tag{8}
$$

Baseline Models: We develop the following baselines: (i) Seq2Seq: This is a basic encoder-decoder (Sutskever et al., 2014) framework with no persona, sentiment and emotion information. (ii). HRED: A general hierarchical encoder-decoder framework (Serban et al., 2017) that captures the conversational context without the persona, sentiment and emotion information. (iii). Seq2Seq + E + P: The utterance encoder along with persona encoder and emotion information is used to decode the responses in a similar manner as (Firdaus et al., 2020). (iv). HRED + E + P: We infuse the persona and emotion in the basic hierarchical encoder-decoder framework. (v). Seq2Seq + E + P + S: The utterance encoder along with persona encoder, sentiment information and emotion information is used to decode the responses. (vi). HRED + E + P + S: We infuse the persona, sentiment and emotion in the basic hierarchical encoder-decoder framework. (vii) Trans: Basic transformer network without persona, sentiment and emotion information. (viii) Trans + E + P: The transformer encoders along with persona encoder and emotion information is used to generate the responses.

4 Dataset and Experimentation

Dataset Description: On the recently published ConvAI2 benchmark dataset, which is an extended version (with a new test set) of the persona-chat dataset (Zhang et al., 2018), we conduct our experiments. The interactions are collected from the randomly paired crowd workers who were instructed to play the part of a given persona. In over 10,981 dialogues, this dataset comprises of 164,356 utterances and has a collection of 1,155 personas, each consisting of at least four personality texts. There are 1,016 dialogues in the testing set and 200 never before seen personas. As the dataset is not labeled with emotions, we use the emotion annotated version of the dataset used in (Firdaus et al., 2020).

Dataset Preparation: As sentiment and emotions are highly co-related we annotate the PersonaChat dataset using the emotion information in a similar manner as (Poria et al., 2019). As emotions such as *excited, grateful, joyful, caring, hopeful, faithful, impressed* have a positive undertone hence we automatically label the utterances having these emotion labels as positive sentiment. Similarly for emotions such as *angry, sad, annoyed, disgusted, terrified, furious, disappointed, jealous* has a negative undertone hence are labelled as negative sentiment. For the other emotion labels such as *surprise, proud, nostalgic, guilty, confident, prepared, sentimental* that can either be positive, neutral or negative depending on the utterance and the context we resort to manual annotation. For annotating the utterances in the PersonaChat dataset, we employ four graduate students highly proficient in English comprehension. The guidelines for annotation along with some examples were explained to the annotators before starting the annotation process. Majority voting scheme was used for selecting the final sentiment label for each utterance. We achieve an overall Fleiss' (Fleiss, 1971) kappa score of 0.75 for sentiment which can be considered as reliable. Detailed statistics of the PersonaChat dataset are provided in Table 2.

Implementation Details: All the implementa-

Dataset Statistics	Train	Valid	Test
Dialogues	7686	1640	1655
Utterances	124816	19680	19860
Avg. turns per Dialogue	12.51	12.73	12.74
Avg. words in a Response	11.89	9.57	10.75
Emotions per dialogue	7.4	6.5	5.1
Unique words	20322	13415	15781

Table 2: Statistics of the PersonaChat Dataset

tions are done using the PyTorch[1] framework. For all the models, including baselines, the batch size is set to 32. We use the dropout (Srivastava et al., 2014) with probability 0.45. During decoding, we use a beam search with beam size 10. We initialize the model parameters randomly using a Gaussian distribution with the Xavier scheme (Glorot and Bengio, 2010). We employ AMSGrad (Reddi et al., 2019) as the optimizer for model training to mitigate the slow convergence issues. We use uniform label smoothing with $\epsilon = 0.1$ and perform gradient clipping when the gradient norm is over 5. To reduce data sparsity, all the numbers and names are replaced with <number> and <person>.

Automatic Evaluation Metrics: In order to assess the model at the emotional and grammatical level, we present the results using the traditional automatic metrics. Perplexity(Chen et al., 1998) is stated to test our proposed framework at the content level. We also report the results using the standard metrics like BLEU-4 (Papineni et al., 2002) and Rouge-L (Lin, 2004) to measure the ability of the generated response for capturing the correct information. BLEU measures the n-grams overlap between the generated response and the gold response, and has become a standard measure for comparing task-oriented dialog systems. It is used to measure the content preservation in the generated responses. We report Distinct-1 and Distinct-2 metrics that measure the distinct n-grams in the generated responses and are scaled with respect to the total number of generated tokens to avoid repetitive and boring responses (Li et al., 2016b). To measure the emotional content in the generated responses, we calculate the emotion accuracy using the pre-trained BERT classifier on the responses generated by the baseline and proposed models.

Human Evaluation Metrics: We randomly sample 500 responses from the test set for human evaluation. For a given input along with persona information, six annotators with post-graduate exposure were assigned to evaluate the quality of the generated responses by the different approaches in a similar manner as the existing works (Firdaus et al., 2020). First, we evaluate the quality of the response on two conventional criteria: *Fluency*, and *Relevance*. These are rated on a five-scale, where 1, 3, 5 indicate unacceptable, moderate, and excellent performance, respectively, while 2 and 4 are used for unsure. Secondly, we evaluate the persona, sentiment and emotion inclusion in response in terms of *Persona Consistency* metric, *Sentiment Coherence* metric and *Emotion Appropriateness* to judge whether the response generated is in consonance to the specified persona, sentiment and the emotion is also coherent to the conversational history. In the case of all these metrics, 0 indicates an irrelevant or contradictory persona, sentiment or emotion in the response, and 1 represents the consistent response to the specified persona, sentiment and emotion. For the human evaluation metrics, we calculate the Fleiss' kappa (Fleiss, 1971) to determine the inter-rater consistency. For fluency and relevance, the kappa score is 0.75, and for emotion appropriateness, sentiment coherence and persona consistency, these are 0.75, 0.71 and 0.78, respectively, indicating "substantial agreement".

5 Result and Analysis

For thorough analysis of our proposed framework, we provide a detailed analysis of the results (both automatic and manual) along with the generated responses. We also analyze the errors made by the network in generating empathetic personalized responses.

Automatic Evaluation Results: The automatic evaluation results are presented in Table 3, which demonstrates that the proposed framework significantly outperforms all the baselines with respect to all the metrics. The final proposed transformer network shows a notable drop in perplexity scores, thereby ensuring grammatically correct responses generated by the framework. In addition, we see that the BLEU scores have increased with an improvement of more than 5% from the basic Seq2Seq framework and with a gain of 4% from the typical HRED model. By introducing the persona and emotion information in the basic Seq2Seq and HRED model, we see the growth in performance, establishing the need for persona and emotion knowledge for generating empathetic, personalized responses. Similarly, in the case of Rouge-L,

[1] https://pytorch.org/

Model Description		Perplexity	BLEU-4	Rouge-L	Distinct-1	Distinct-2	Emotion Accuracy
Baseline Approaches	***Seq2Seq*** *(Sutskever et al., 2014)*	56.11	0.089	0.196	0.0125	0.0464	0.358
	HRED *(Serban et al., 2017)*	55.63	0.096	0.201	0.0128	0.0469	0.376
	Seq2Seq + E + P *(Firdaus et al., 2020)*	54.13	0.103	0.189	0.0168	0.0549	0.657
	HRED + E + P	54.85	0.116	0.224	0.0174	0.0592	0.665
	Seq2Seq + E + P + S	53.61	0.115	0.203	0.0171	0.0555	0.673
	HRED + E + P + S	52.46	0.127	0.237	0.0186	0.0590	0.689
Proposed Approach	***Trans + E + P + S***	**51.92**	**0.143**	**0.266**	**0.0219**	**0.0987**	**0.715**
Ablation Study	***Trans***	53.47	0.118	0.239	0.0189	0.0883	0.678
	Trans + E + P	53.44	0.125	0.242	0.0193	0.0896	0.695

Table 3: Results of automatic evaluation. Here, E-Emotion, P-Persona, S-Sentiment, Trans-Transformers

Model Description		Fluency	Relevance	Emotion Appropriateness	Persona Consistency	Sentiment Coherence
Baseline Approaches	***Seq2Seq*** *(Sutskever et al., 2014)*	2.98	2.65	38%	35%	33%
	HRED *(Serban et al., 2017)*	3.16	2.89	41%	39%	37%
	Seq2Seq + E + P *(Firdaus et al., 2020)*	3.29	3.02	53%	48%	42%
	HRED + E + P	3.45	3.18	59%	55%	47%
	Seq2Seq + E + P + S	3.33	3.21	56%	52%	51%
	HRED + E + P + S	3.52	3.45	61%	58%	55%
Proposed Approach	***Trans + E + P + S***	**3.89**	**3.66**	**65%**	**67%**	**62%**

Table 4: Results of human evaluation for the existing baselines and the proposed framework

Dialog Input	Persona Information		Ground-Truth	Generated Response
[Person 1] What do you do for a career? [Person 2] I like to watch kids. [Person 1] I actually play guitar and do lot of welding.	Persona 1 I am primarily a meat eater. I am a guitar player. Welding is my career field. My parents don't know I am gay.	Persona 2 I've a sweet tooth. I'm a babysitter and drive a mercedes. I'm the middle child of 3 sibling. I'm getting married in six weeks.	What do you weld houses? Neutral	**Seq2Seq:** What is weld? ¡UNK¿ ¡UNK¿ **HRED:** What you weld? **Seq2Seq+E+P:** Wow! What you weld mostly? (surprise) **HRED+E+P:** Really! Do you weld houses? (surprise) **Proposed:** Wow that's great, what do you weld mostly, is it houses? (surprise)
[Person 1] Hi! do you like turtles? [Person 2] I have two cats actually. [Person 1] I have a turtle his name is Speedy.	Persona 1 I don't pick up my toys. I have a pet turtle. I like to play with my dolls. I'm fourteen.	Persona 2 I love cats and have two cats. My favorite season is winter. I won a gold medal in the 1992 olympics. I've a hat collection of over 1000 hats.	I am a cat person. Positive	**Seq2Seq:** I have cats. **HRED:** I like cats mostly. **Seq2Seq+E+P:** Turtles are nice but I like cats. (joy) **HRED+E+P:** Nice name for a pet, but I love cats. (joy) **Proposed:** That is an adorable name for a turtle! but I am a cat person. (joy)

Table 5: Examples of responses generated by different models having emotion, sentiment and persona

we see a remarkable improvement in the performance of the proposed network compared to the *Seq2Seq+E+P* and *HRED+E+P* frameworks, respectively. We also report the emotion accuracy of the generated response. It is quite apparent that the responses having emotion information have higher accuracy than the models with no emotion information. The proposed model outperforms the best performing baseline network with an improvement of 5% in emotion accuracy.

We also include the results of distinct-1 and distinct-2 to demonstrate that the responses generated are varied and diversified. From assessment, it is clear that the proposed system is also competent enough to make the response diverse and interactive, along with producing emotional and persona-guided responses. By including the sentiment information in the contextual history, we see that there is improvement in the proposed framework as it facilitates in generating the correct emotional responses in accordance to the sentiment of the speaker.

We also perform an ablation study on the proposed framework to understand the importance of the emotion, persona and sentiment information in enhancing the performance of the overall framework. It is evident that the proposed framework compared to *Trans* and *Trans + E + P* models performs better as it also includes the sentiment of the previous utterances proving the significance of all the three components in generating better and interactive responses.

Human Evaluation Results: The manual evaluation, the results of which are recorded in Table 4, is carried out for a more comprehensive analysis of our proposed system. From the table, it is clear that the proposed system provides better performance with regards to all the specified metrics than

the existing approaches. As fluency calculates the grammatical accuracy of the response generated, it can, therefore, be assumed that the proposed model generates fluent responses. The final model having the highest scores in the case of fluency, as opposed to the baseline system, proves that the responses are grammatically correct and complete. Similarly, in the case of emotional content in the responses, we see that the frameworks having the emotion information seem to generate empathetic responses instead of the basic Seq2Seq and HRED frameworks. With an improvement of 6%, the proposed network surpasses the emotion score of the *HRED+E+P* model.

We also compute the ability of the models to maintain a consistent persona while generating the responses. From the manual evaluation results presented in the table, we can see that the *HRED+E+P* and *Seq2Seq+E+P* models show significant improvement from the typical HRED and Seq2Seq model in inducing the persona information while generating the responses. There is an enhancement in the proposed system from all of the other baseline systems in the case of the persona consistency metric. Also, the sentiment coherence score of the proposed framework is higher in comparison to the models without the sentiment information marking the importance of sentiment in the overall framework.

Through human assessment, it can, therefore, be inferred that the proposed system is capable of producing empathetic responses and has the capacity to retain a particular persona and respond with the correct sentiment.

Case Study and Discussion: In Table 5, we provide two examples and their corresponding generated responses by the different models. For both the examples, it is evident that the basic Seq2Seq and HRED frameworks generate short and non-emotional responses that do not increase user engagement. On the contrary, the baseline models having the knowledge of both persona and emotion generate empathetic and personalized responses. Moreover, the responses generated by our proposed framework are not only fluent but also are engaging, diverse, personalized, and emotionally appropriate to the conversational history and the sentiment.

We came across through some of the errors made by the baseline and proposed frameworks after performing a detailed comparative analysis of the generated responses. Some of the commonly occurring errors are: **(i) Extra information:** There are a few instances where the information in the input is found to be replicated, and extra words added, in both the baselines and the proposed system providing extra information. For example, Gold: *if I have time for cooking and repairing houses*; Predicted: *if I have time time hunting, cooking...* **(ii) Persona Discrepancy:** For certain instances, the responses generated by the proposed architecture are incompatible with the personality information and lack the precise details present in the speaker's persona texts. For example, Persona information: *I have 3 lovely kids and enjoy playing with them.* Predicted response: *I hate kids find them very annoying.*

6 Conclusion and Future Work

Grounding conversations based on the user's persona and emotions is an exciting research direction that can contribute to building natural, engaging and social conversational agents. Our current work presents one of the first examples of an empathetic, personalized dialogue generation for building a robust social chatbot using the sentiment information of the speaker in the ongoing conversation. We trained a novel transformer framework capable of generating responses that is sensitive to the emotions of the speaker and in accordance with their persona information and sentiment. Specifically, the experimental analysis on the PersonaChat dataset shows that the responses having both the emotional quotient and persona knowledge in the responses help build interactive and engaging conversations.

Our future work would focus on extending the architectural framework for improving the performance of the generation. In addition, we would also investigate other factors such as politeness, diversity in responses for creating a comprehensive social chatbot.

7 Ethical Declarations

All the resources used in this paper are publicly available. The dataset used in this paper is used only for the purpose of academic research. There is nothing to disclose that warrant the ethical issues.

Acknowledgement

Authors duly acknowledge the support from the Project titled "Sevak-An Intelligent Indian Language Chatbot", Sponsored by SERB, Govt. of India (IMP/2018/002072). Asif Ekbal acknowledges

the Young Faculty Research Fellowship (YFRF), supported by Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Elia Bruni and Raquel Fernandez. 2017. Adversarial evaluation for open-domain dialogue generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 284–288.

Stanley Chen, Douglas H. Beeferman, and Ronald Rosenfeld. 1998. Evaluation metrics for language models.

Pierre Colombo, Wojciech Witon, Ashutosh Modi, James Kennedy, and Mubbasir Kapadia. 2019. Affect-driven dialog generation. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3734–3743.

Mauajama Firdaus, Naveen Thangavelu, Asif Ekba, and Pushpak Bhattacharyya. 2020. Persona aware response generation with emotions. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-lm: A neural language model for customizable affective text generation. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 634–642.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 249–256.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778.

Chenyang Huang, Osmar R Zaiane, Amine Trabelsi, and Nouha Dziri. 2018. Automatic dialogue generation with expressed emotions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 49–54.

Chaitanya K Joshi, Fei Mi, and Boi Faltings. 2017. Personalization in goal-oriented dialog. *arXiv preprint arXiv:1706.07503*.

Lee Kezar. 2018. Mixed feelings: Natural text generation with variable, coexistent affective categories. In *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018,Student Research Workshop*, pages 141–145.

Jingyuan Li and Xiao Sun. 2018. A syntactically constrained bidirectional-asynchronous approach for emotional conversation generation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 678–683.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016a. A persona-based neural conversation model. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Jiwei Li, Will Monroe, Alan Ritter, Daniel Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1192–1202.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 986–995.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004, Barcelona, Spain*.

Zhaojiang Lin, Peng Xu, Genta Indra Winata, Zihan Liu, and Pascale Fung. 2019. Caire: An end-to-end empathetic chatbot. *arXiv preprint arXiv:1907.12108*.

Liangchen Luo, Wenhao Huang, Qi Zeng, Zaiqing Nie, and Xu Sun. 2019. Learning personalized end-to-end goal-oriented dialog. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI*

Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, volume 33, pages 6794–6801.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *EMNLP*.

Andrea Madotto, Zhaojiang Lin, Chien-Sheng Wu, and Pascale Fung. 2019. Personalizing dialogue agents via meta-learning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5454–5459.

Bilyana Martinovski and David Traum. 2003. Breakdown in human-machine interaction: the error is the clue. In *Proceedings of the ISCA tutorial and research workshop on Error handling in dialogue systems*, pages 11–16.

Abraham Harold Maslow. 1943. A theory of human motivation. *Psychological review*, 50(4):370.

Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018. Training millions of personalized dialogue agents. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2775–2779.

Myriam D Munezero, Calkin Suero Montero, Erkki Sutinen, and John Pajunen. 2014. Are they different? affect, feeling, emotion, sentiment, and opinion detection in text. *IEEE transactions on Affective Computing*, 5(2):101–111.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. Association for Computational Linguistics.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2019. Meld: A multimodal multi-party dataset for emotion recognition in conversations. *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 527–536.

Helmut Prendinger, Junichiro Mori, and Mitsuru Ishizuka. 2005. Using human physiology to evaluate subtle expressivity of a virtual quizmaster in a mathematical game. *International journal of human-computer studies*, 62(2):231–245.

Qiao Qian, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2017. Assigning personality/identity to a chatting machine for coherent conversation generation. *IJCAI*.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5370–5381.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2019. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*.

Ruhi Sarikaya. 2017. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine*, 34(1):67–81.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3295–3301.

Heung-Yeung Shum, Xiao-dong He, and Di Li. 2018. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1):10–26.

Haoyu Song, Wei-Nan Zhang, Yiming Cui, Dong Wang, and Ting Liu. 2019. Exploiting persona information for diverse generation of conversational responses. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5190–5196.

Haoyu Song, Wei-Nan Zhang, Jingwen Hu, and Ting Liu. 2020. Generating persona consistent dialogues by exploiting natural language inference. *AAAI*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

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

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Bowen Wu, Mengyuan Li, Zongsheng Wang, Yifu Chen, Derek Wong, Qihang Feng, Junhong Huang, and Baoxun Wang. 2020. Guiding variational response generator to exploit persona. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 53–65.

Xianchao Wu, Ander Martinez, and Momo Klyen. 2018. Dialog generation using multi-turn reasoning neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, volume 1, pages 2049–2059.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3940–3949.

Zhen Xu, Bingquan Liu, Baoxun Wang, Cheng-Jie Sun, Xiaolong Wang, Zhuoran Wang, and Chao Qi. 2017. Neural response generation via gan with an approximate embedding layer. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 617–626.

Semih Yavuz, Abhinav Rastogi, Guan-Lin Chao, and Dilek Hakkani-Tur. 2019. Deepcopy: Grounded response generation with hierarchical pointer networks. *NeurIPS*.

Jiayi Zhang, Chongyang Tao, Zhenjing Xu, Qiaojing Xie, Wei Chen, and Rui Yan. 2019. Ensemblegan: Adversarial learning for retrieval-generation ensemble model on short-text conversation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 435–444.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2204–2213.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*.

Xianda Zhou and William Yang Wang. 2017. Mojitalk: Generating emotional responses at scale. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1128–1137.

Qingfu Zhu, Lei Cui, Weinan Zhang, Furu Wei, and Ting Liu. 2019. Retrieval-enhanced adversarial training for neural response generation. *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3763–3773.

Biomedical Data-to-Text Generation via Fine-Tuning Transformers

Ruslan Yermakov
Decision Science
& Advanced Analytics
Bayer AG
yermakovruslan@gmail.com

Nicholas Drago
Regulatory Policy
and Intelligence
Bayer AG

Angelo Ziletti*
Decision Science
& Advanced Analytics
Bayer AG
angelo.ziletti@bayer.com

Abstract

Data-to-text (D2T) generation in the biomedical domain is a promising - yet mostly unexplored - field of research. Here, we apply neural models for D2T generation to a real-world dataset consisting of package leaflets of European medicines. We show that fine-tuned transformers are able to generate realistic, multi-sentence text from data in the biomedical domain, yet have important limitations. We also release a new dataset (*BioLeaflets*) for benchmarking D2T generation models in the biomedical domain.

1 Introduction

Data-to-text (D2T) systems are attracting considerable interest due to their ability to automate the time-consuming writing of data-driven reports. There is a hitherto largely untapped potential for text generation in the biomedical domain. Potential applications of natural language generation of patient-friendly biomedical text include preparation of the first draft of package leaflets, patient education materials, or direct-to-consumer promotional materials in countries where this is permitted. Here we focus on a D2T task aiming to generate fluent and fact-based descriptions from biomedical data.

2 Related Work

Recently, neural D2T models have significantly improved the quality of short text generation (usually one sentence long) from input data compared to multi-stage pipelined or template-based approaches. Examples include biographies from Wikipedia fact tables (Lebret et al., 2016), restaurant descriptions from meaning representations (Novikova et al., 2017b), and basketball game summaries from statistical tables (Wiseman et al., 2017). Still, neural D2T approaches have major challenges, as outlined by Wiseman et al. (2017) and Parikh et al. (2020) which hinder their application to many real-world applications. These include hallucination effects (generated phrases not supported or contradictory to the source data), missing facts (generated text does not include input information), intersentence incoherence, and repetitiveness in the generated text. Following the success of leveraging pre-trained large-scale language models for a large variety of tasks, Kale and Rastogi (2020) fine-tuned T5 models (Raffel et al., 2020) for D2T generation. This strategy achieved state-of-the-art performance on task-oriented dialogue (MultiWoz) (Budzianowski et al., 2018), tables-to-text (ToTTo) (Parikh et al., 2020) and graph-to-text (WebNLG) (Gardent et al., 2017).

To the best of our knowledge, recent neural approaches and transfer learning strategies have not been applied to multi-sentence generation from input data, nor have they been applied in the biomedical domain. Our contribution is two-fold: we introduce a real-world biomedical dataset *BioLeaflets*, and demonstrate that transformers can generate high-quality multi-sentence text from data in the biomedical domain. The *BioLeaflets* dataset, fine-tuned models, code, and generated samples are available at https://github.com/bayer-science-for-a-better-life/data2text-bioleaflets.

3 The *BioLeaflets* Dataset

We introduce a new biomedical dataset for D2T generation - *BioLeaflets*, a corpus of 1336 package leaflets of medicines authorised in Europe, which we obtain by scraping the European Medicines Agency (EMA) website. This dataset comprises the large majority ($\sim$ 90%) of medicinal products authorised through the centralised procedure in Europe as of January 2021.

Package leaflets are published for medicinal products approved in the European Union (EU). They are included in the packaging of medicinal

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 364–370,
Aberdeen, Scotland, UK, 20-24 September 2021.

(a) Original section content	**novonorm** is **an oral antidiabetic medicine** containing **repaglinide** which helps your **pancreas** produce more **insulin** and thereby lower **your blood sugar** (**glucose**). **type 2 diabetes** is **a disease** in which your **pancreas** does not make enough **insulin** to control **the sugar** in your blood or where your **body** does not respond normally to the **insulin** it produces. **novonorm** is used to control **type 2 diabetes** in adults as an add-on to diet and exercise: **treatment** is usually started if diet, **exercise and weight reduction** alone have not been able to control (or lower) **your blood sugar**. **novonorm** can also be given with **metformin**, another medicine for **diabetes**. **novonorm** has been shown to lower **the blood sugar**, which helps to prevent **complications** from **your diabetes**.
(b) Input: entities as a flat string	<PRODUCT_NAME> novonorm </PRODUCT_NAME> <TREATMENT> an_oral_antidiabetic_medicine </TREATMENT> <GENERIC_NAME> repaglinide </GENERIC_NAME> <SYSTEM_ORGAN_SITE> pancreas </SYSTEM_ORGAN_SITE> <GENERIC_NAME> insulin </GENERIC_NAME> <TEST> your_blood_sugar </TEST> ...<TEST> the_blood_sugar </TEST> <DX_NAME> complications </DX_NAME> <PROBLEM> your_diabetes </PROBLEM>
(c) T5 generated section	**novonorm** is **an oral antidiabetic medicine** containing **repaglinide** which helps your **pancreas** produce more **insulin** and thereby lower **your blood sugar** (**glucose**). **type 2 diabetes** is **a disease** in which your panccreases does not make enough **insulin** to control **the sugar** in your blood or where your **body** does not respond normally to the **insulin** it produces. novonourm is used to control type 2 diabetic patients who still have their **diabetes** and who are still unable to respond normally despite **treatment**, **exercise and weight reduction**, which have not helped to control your blood **glucose**. there is also a test that novonorgm can be given in combination with **metformin**, another medicine for **diabetes**. it is important that you continue to follow the advice on diet, exercising and diet that your nurse or doctor has given you.
(d) BART generated section	novondorm is **an oral antidiabetic medicine** containing **repaglinide** which helps your **pancreas** produce more **insulin** and thereby lower **your blood sugar** (**glucose**). **type 2 diabetes** is **a disease** in which your pancrea does not make enough **insulin** to control **the sugar** in your blood or where your **body** does not respond normally to the **insulin** it produces. **novonorm** is used to treat type 2 diabetic in adults. **treatment** is usually started with diet and **exercise and weight reduction**. your blood **glucose** may be increased when you start to take novonor on its own or in combination with **metformin**. if you have **diabetes**, no
(e) Content planner generated section	**novonorm** contains the active substance **repaglinide** which helps to lower **your blood sugar** (**glucose**). **type 2 diabetes** is **a disease** where your **body** does not make enough **insulin** to control **the sugar** in your blood or where your **body** does not respond normally to the **insulin** it produces. repaglinide krka is used to control **type 2 diabetes** in adults as **type 2 diabetes**. **type 2 diabetes** is also called non - insulin - dependent diabetes mellitus. **type 2 diabetes** is also a condition in which your **body** does not make enough **insulin** or the **insulin** that your **body** produces does not work as well as it should. your **body** can also make too much sugar. when this happens, sugar (**glucose**) builds up in the blood. this can lead to serious medical problems like heart disease, kidney disease, 2 and 2.

Table 1: Example of text generations. Entities are highlighted in bold, typos are underlined, and hallucinations are shown in red.

products and contain information to help patients use the product safely and appropriately, under the guidance of their healthcare professional. Package leaflets are required to be written in a way that is clear and understandable (EU, 2001). Each document contains six sections (see Table 2).The main challenges of this dataset for D2T generation are multi-sentence and multi-section target text, small sample size, specialized medical vocabulary and syntax.

3.1 Dataset Construction

The content of each section is not standardized, yet it is still well-structured. Thus, we identify sections via heuristics such as regular expressions and word overlap. The content of each section is lower-cased and tokenized by treating all special characters as separate tokens. Duplicates are also removed. We randomly split the dataset into training (80%), development (10%), and test (10%) set. Table 2 summarizes dataset statistics.

3.2 Dataset Annotations

We do not have annotations available for the package leaflet text. To create the required input for D2T generation, we augment each document by leveraging named entity recognition (NER). Parikh et al. (2020) indicated it is important that target summaries contain information that can be inferred from the input data to avoid dataset-induced hallucinations. To this end, we combine two NER frameworks: Amazon Comprehend Medical (ACM) (Bhatia et al., 2019) and Stanford Stanza (Qi et al., 2020; Zhang et al., 2021). ACM and Stanza achieved entity micro-averaged test F1 of 85.5% and 88.13% respectively on the 2010 i2b2/VA clinical dataset (Uzuner et al., 2011). We further leverage ACM to detect medical conditions from ICD-10 (WHO, 2004) and medications from RxNorm. Additionally, we treat all digits as entities, and add the medicine name as first entity. In case of overlapping entities from different sources, we favor longer entities over shorter ones. As a result of the NER

Section type	No. samples	Average length (characters)	Average length (tokens)	Average no. entities per section	No. unique entities
1. What the product is and what it is used for	1 314	963	174	29.3	9 641
2. What you need to know before you take the product	1 309	4 560	849	127.7	23 278
3. How to take the product	1 313	2 300	458	50.5	11 640
4. Possible side effects	1 295	3 453	651	135.2	27 945
5. How to store the product	1 172	631	123	6.3	2 041
6. Content of the pack and other information	1 311	982	196	38.4	9 932

Table 2: *BioLeaflets* dataset statistics grouped by section type.

process, we obtain 26 unique entity types. Examples are: *problem*: ('active chronic hepatitis', 'migraine pain'), *system-organ-site*: ('blood vessel', 'kidneys', 'surrounding tissue'), *treatment*: ('routine dental care', 'a vaccination', 'a chemotherapy medicinal product'), or *procedure*: ('injections', 'spinal or epidural anaesthesia', 'surgical intervention', 'bone marrow or stem cell transplant').

BioLeaflets proposes a conditional generation task: given an ordered set of entities as source, the goal is to produce a multi-sentence section. Since only the entities are provided as input, the structured data is underspecified. A human without specialized knowledge would likely be unable to produce satisfactory text. However, we expect that a labeling expert with profound knowledge of package leaflets would be able to generate (with some difficulty) satisfactory text in the large majority of cases. Successful generation thus requires the model to learn specific syntax, terminology, and writing style from the corpus (e.g., via fine-tuning).

4 Experiments

Following Kale and Rastogi (2020), we represent the structured data (i.e., detected entities) as a flat string (linearization). The entities are kept in their order of appearance (Table1b). The models are then trained to predict - starting from these entities - the corresponding published leaflet text.

We present baseline results on *BioLeaflets* dataset by employing the following state-of-the-art approaches:

- **Content Planner**: two stages neural architecture (content selection and planning) based on LSTM (Puduppully et al., 2019). Since only relevant entities are provided as input to the model, we solely use the content planning stage (encoder-decoder architecture with an attention mechanism). We train one model for each section, and use the same hyperparameters reported by Puduppully et al. (2019).

- **T5**: a text-to-text transfer transformer model (Raffel et al., 2020). Kale and Rastogi (2020): showed that T5 outperforms alternatives like BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019). After hyperparameter search on the development dataset, the following parameters (yielding the best ROUGE-L score (Lin, 2004)) are selected: constant learning rate of 0.001, batch size of 32, 20 epochs, greedy search as a decoding method.

- **BART**: denoising autoencoder for pretraining sequence-to-sequence models with transformers (Lewis et al., 2020). For computational reasons, we use the same hyperparameters as per T5 fine-tuning.

- **BART and T5 with conditioning**: we add the prefix "section_n" ($n = 1, \ldots 6$) to the (linearized) input data. This explicitly gives the model information on the section number and thus enforces a conditioning on the section type for text generation.

BART and T5 fine-tuning are performed via HuggingFace (Wolf et al., 2020).

5 Evaluation

Table 1 shows the generated text for one test sample as illustrative example. All generated text is

Model	Word-overlap metrics		Semantic equivalence metrics		
	SacreBLEU	ROUGE-L	BERTScore	BLEURT	MoverScore-2l
Content Planner	**27.78**	39.32	0.214	-0.072	0.591
BART-base	8.76 ± 0.02	42.73 ± 0.11	**0.370** ± 0.001	**0.268** ± 0.002	0.609 ± 0.0004
BART-base + cond	8.73 ± 0.02	42.60 ± 0.12	**0.369** ± 0.001	**0.268** ± 0.003	0.608 ± 0.0004
T5-base	18.68 ± 0.07	**47.22** ± 0.17	0.363 ± 0.001	0.255 ± 0.008	**0.620** ± 0.0005
T5-base + cond	18.63 ± 0.14	**47.31** ± 0.22	0.364 ± 0.002	0.256 ± 0.006	**0.621** ± 0.0008

Table 3: Results on the BioLeaflets test set (averaged over all sections). T5 and BART models are fine-tuned with seven different random seeds: average and standard deviation are reported. BLEURT-large-128 is used.

Model		Adequacy	Hallucination presence	Entity inclusion	Fluency
Content Planner	annotator 1	4.1 ± 3.0	6.8 ± 3.2	4.8 ± 3.2	5.1 ± 3.3
	annotator 2	3.7 ± 2.6	6.4 ± 2.5	5.1 ± 2.5	5.4 ± 2.3
BART-base	annotator 1	7.5 ± 2.1	3.1 ± 2.6	7.4 ± 2.3	8.6 ± 1.8
	annotator 2	**6.6** ± 2.2	**3.3** ± 2.1	**8.1** ± 1.8	8.0 ± 1.3
T5-base	annotator 1	**7.8** ± 1.8	**3.0** ± 2.4	**7.6** ± 2.1	**9.0** ± 1.4
	annotator 2	6.5 ± 2.2	3.5 ± 1.9	7.8 ± 1.7	**8.2** ± 1.2

Table 4: Human evaluation of test samples. Values on a scale from one to ten; average and standard deviation are reported. The higher the better for all quantities, expect for "Hallucination presence". Adequacy estimates the overall generation quality, taking into consideration fluency, amount of hallucination, and entities included in the generated text.

made available[1]. After a thorough inspection of the samples, we conclude that generated text is generally fluent and coherent. Text produced by T5 and BART is more fluent, factually and grammatically correct than those by Content Planner. Table 3 illustrates the performance of state-of-the-art models quantified by automatic metrics. Word-overlap metrics such as (Sacre)BLUE (Post, 2018) and ROUGE (Lin, 2004) have been shown to perform poorly in evaluation of natural language generation (Novikova et al., 2017a), and thus we report them here only for completeness. Conversely, contextual embedding based metrics BERTScore (Zhang* et al., 2020), BLEURT (Sellam et al., 2020), and MoverScore-2 (Zhao et al., 2019) correlate with human judgment on sentence-level and system-level evaluation. They adequately capture semantic equivalence between generated and target text as well as fluency and overall quality. T5 and BART outperform Content Planner, as measured by BERTscore, BLEURT, and MoverScore-2. T5 and BART show similar performance. These results show that transformer-based models and transfer learning strategies achieve state-of-the-art performance on data-to-text tasks, generalizing the findings in Kale and Rastogi (2020) to multi-sentence and multi-section generation, biomedical text, and low-data setting.

To confirm these findings, human evaluation is performed for Section 1 of the test set by two annotators. Results are shown in Table 4. Similarly to Manning et al. (2020), we design a survey which includes adequacy (estimate of overall quality), presence of hallucinations, entity inclusion, and fluency. T5 and BART have similar performance, and they produce more adequate text than Content Planner. T5 and BART performance is more stable across samples (lower standard deviation). These conclusions coincide with the ones drawn from Table 3, thus confirming the usefulness of semantic equivalence metrics for automatic evaluation of text generation.

Interestingly, specifying the section type in the input records (i.e., explicit conditioning) did not improve model performances (Table 3). To rationalize this result, we analyze T5 internal representations. Specifically, for each test sample, we extract the (average) last encoder hidden-state for both pre-trained (not fine-tuned) and fine-tuned T5 (fine-tuned on *BioLeaflets* but without explicit

[1] https://github.com/bayer-science-for-a-better-life/data2text-bioleaflets

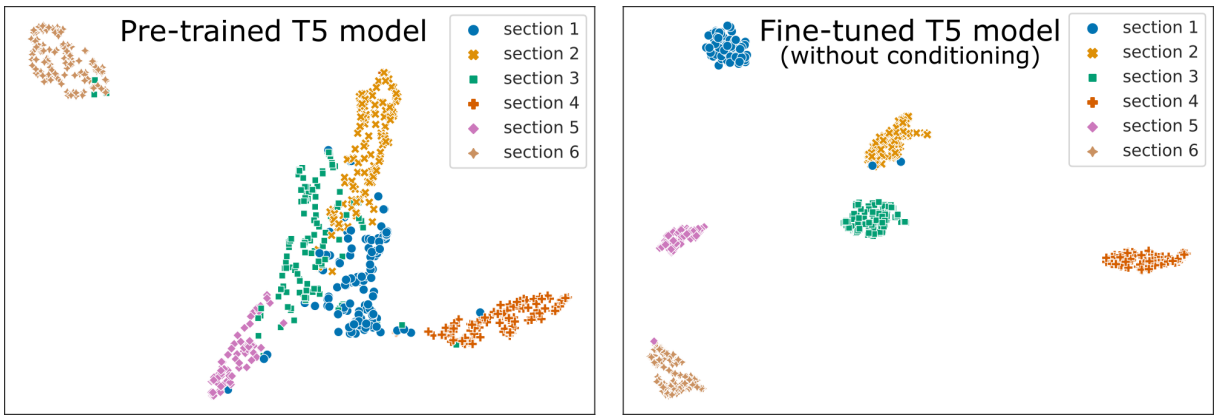


Figure 1: Two-dimensional projections of T5 internal representations (average of the last encoder hidden-states) for pre-trained (not fine-tuned) (**left**) and fine-tuned T5 model on *BioLeaflets* dataset (**right**). T5 implicitly learns to condition on section type during fine-tuning.

conditioning). We then project these vectors into two-dimensions using the non-linear dimensionality reduction method UMAP (McInnes et al., 2020). The results are depicted in Fig. 1. In Fig. 1 (right), we can identify six well-separated clusters, which correspond to (the internal representations of samples belonging to) the six document sections in the *BioLeaflets* dataset. Thus, after fine-tuning, T5 maps input data belonging to different sections to different parts of the internal representation space. The cluster separation is much less pronounced for the pre-trained (not-fine-tuned) T5 model (Fig. 1, left). This shows that during the fine-tuning process, T5 implicitly learns to condition on section type, thus learning to generate different sections, even despite the small dataset. Since conditioning is learned automatically, explicitly passing the section type as input does not increase model performance.

6 Error Analysis and Limitations

After thorough qualitative evaluation of numerous generated samples, the following general issues appear:

- **Typos**: Even though models largely utilize the input entities correctly, typos appear in generated text by T5 and BART for out-of-vocabulary words, e.g. Table 1 (c, d). Content Planner does not seem to have this problem.

- **Hallucinations** are present for all models. Loss functions like maximum likelihood do not directly minimize hallucinations, thus hindering consistent fact-based text generation.

- **Repetitiveness**: Content Planner produce repetitions (e.g. Table 1 (e)), whereas T5 and BART language models do not.

- **Difficulties in producing coherent long text**: In the *BioLeaflets* dataset, models perform well in generating section 1, which is 962 characters long on average. However, the quality of section 4 "Possible side effects" (3 453 characters long on average) generation is poor.

Possible improvements to our work are: analysis of the impact of shuffling of entities for the input data generation, introduction of loss functions that explicitly favor factual correctness, usage of specialized biomedical embeddings, inclusion of more source input data (e.g. part-of-speech, dependency tag), generation of longer text (beyond the 512 tokens generated here).

7 Conclusion

In this study, we introduce a new biomedical dataset (*BioLeaflets*), which could serve as a benchmark for biomedical text generation models. We demonstrate the feasibility of generating coherent multi-sentence biomedical text using patient-friendly language, based on input consisting of biomedical entities. These results show the potential of text generation for real-world biomedical applications. Nevertheless, human evaluation is still a required step to validate the generated samples. Application of the methodology and models used here to different sets of biomedical text (e.g., generation of selected sections of clinical study reports) could be an area for further research.

References

Parminder Bhatia, Busra Celikkaya, Mohammed Khalilia, and Selvan Senthivel. 2019. Comprehend medical: A named entity recognition and relationship extraction web service. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1844–1851.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

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

European Union EU. 2001. Directive 2001/83/ec of the european parliament and of the council of 6 november 2001 on the community code relating to medicinal products for human use. Brussels, Belgium.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Emma Manning, Shira Wein, and Nathan Schneider. 2020. A human evaluation of amr-to-english generation systems. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 4773–4786. International Committee on Computational Linguistics.

Leland McInnes, John Healy, and James Melville. 2020. Umap: Uniform manifold approximation and projection for dimension reduction.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017a. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017b. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1173–1186. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6908–6915.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.

WHO WHO. 2004. Icd-10 : international statistical classification of diseases and related health problems : tenth revision.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yuhao Zhang, Yuhui Zhang, Peng Qi, Christopher D Manning, and Curtis P Langlotz. 2021. Biomedical and clinical English model packages for the Stanza Python NLP library. *Journal of the American Medical Informatics Association*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Decoding, Fast and Slow: A Case Study on Balancing Trade-Offs in Incremental, Character-level Pragmatic Reasoning

Sina Zarrieß[1], Hendrik Buschmeier[1], Ting Han[2], Simeon Schüz[1]
[1]Bielefeld University, [2]Artificial Intelligence Research Center, Tokyo
[1]{sina.zarriess,hbuschme,simeon.schuez}@uni-bielefeld.de,
[2]ting.han@aist.go.jp

Abstract

Recent work has adopted models of pragmatic reasoning for the generation of informative language in, e.g., image captioning. We propose a simple but highly effective relaxation of fully rational decoding, based on an existing incremental and character-level approach to pragmatically informative neural image captioning. We implement a mixed, 'fast' and 'slow', speaker that applies pragmatic reasoning occasionally (only word-initially), while unrolling the language model. In our evaluation, we find that increased informativeness through pragmatic decoding generally lowers quality and, somewhat counter-intuitively, increases repetitiveness in captions. Our mixed speaker, however, achieves a good balance between quality and informativeness.

1 Introduction

Kahneman (2011) famously said that humans have two ways of thinking (along with others theories on human information processing, e.g., Schneider and Shiffrin, 1977): one way is *fast*, automatic and intuitive, the other is a *slow*, controlled, and explicit way of reasoning. This distinction also arises in research on human language processing: slow processes of reasoning that allow speakers to adapt their utterances very flexibly and strategically to a given context are central to theories of pragmatics (Searle, 1969; Grice, 1975; Clark, 1996). Yet, speakers are known to produce utterances in context quickly and easily, which has been a central concern in, e.g., psycholinguistics and experimental pragmatics (Keysar et al., 1998; Galati and Brennan, 2010; Degen and Tanenhaus, 2016, 2019). Similarly, models of pragmatic reasoning and their applications in NLP face the challenge that fully rational language generation is computationally costly or even intractable (Reiter, 1991; White et al., 2020).

Recent work on pragmatics in NLP has taken interest in the Rational Speech Acts (RSA) model (Frank and Goodman, 2012) which resulted in implementations of frameworks that model the generation of informative language with so-called *rational speakers* (Andreas and Klein, 2016; Fried et al., 2018; Shen et al., 2019). Cohn-Gordon et al. (2018) use an image captioning set-up inspired by classical reference games (see Figure 1) to show that a 'slow' rational speaker which reasons internally about the informativeness of utterances generated by a plain neural language model is communicatively more effective than a 'fast' literal speaker that produces the most likely utterance for the target as predicted by the language model. More generally, recent work in NLG has shown a lot of interest in reasoning or decoding methods that extend neural generation models with additional objectives that cannot be easily achieved by decoding the underlying neural language model with greedy or beam search (e.g., Li et al., 2016; Vedantam et al., 2017; Vijayakumar et al., 2018; Ippolito et al., 2019; Holtzman et al., 2020; Tam, 2020).

Reasoning schemes like RSA provide an attractive, since explicit and theoretically motivated, way of incorporating linguistically plausible, communicative strategies into a neural generation framework. At the same time, however, RSA and various related decoding methods have been found to not achieve a good balance between different dimensions of output quality. For instance, Ippolito et al. (2019) investigates a range of decoding methods that aim at increasing the lexical diversity of image captions or responses in dialogue and report on a very clear quality-diversity trade-off: the more the decoding procedure (e.g., sampling) increases diversity and deviates from the predictions of the underlying language model, the more the generated expressions decrease in quality. Recently, Schüz et al. (2021) found similar trade-offs for decod-

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 371–376,
Aberdeen, Scotland, UK, 20-24 September 2021.

S_0 a group of people riding on the backs of horses
S_1 two brown hornes grazing in a fenced grassy field
S_x two horses in a field in front of a field

Figure 1: Captions for the target image (large), generated by a literal (S_0), rational (S_1) and mixed speaker (S_x), with rationality parameter $\alpha = 5$ and beam search. The captions by S_1 and S_x are more discriminative ("field"), but contain repetitions and out-of-vocabulary words ("hornes").

ing word-level image captioning models with RSA and Vedantam et al. (2017)'s discriminative beam search.

Next to these trade-offs, rational speakers in RSA, which apply complex recursive reasoning using an internal listener and speaker, incur a high computational cost, particularly in generation setups with large candidate spaces where exhaustive search is not tractable. Therefore, recent works have implemented incremental decoding schemes that reason about discriminativeness at *every* time-step, during unrolling the language model (Vedantam et al., 2017; Cohn-Gordon et al., 2019). Cohn-Gordon et al. (2018)'s character-level approach fully confronts pragmatic reasoning: the neural language model captions images in a character-by-character fashion such that each character can be internally scored for its informativeness by the rational speaker. While this incremental generation and reasoning scheme makes it possible to search a large space of potential utterances, it is still extremely costly as the internal, recursive reasoning of the speaker is applied at every character.

In this paper, we propose a very simple but highly efficient relaxation of fully rational and incremental decoding with RSA: we propose a *mixed speaker* that switches between literal and rational inference, that is, between 'fast' and 'slow' decoding, while unrolling the language model. This speaker applies pragmatic reasoning at *particular* time steps during decoding rather than at every time-step. Extending Cohn-Gordon et al. (2018)'s character-level RSA, our mixed speaker is rational only when generating the first character of a new word and uses fast literal decoding for the remaining steps in the sequence. Adopting Schüz et al. (2021)'s evaluation setting that combines quality, informativeness and diversity, we find that Cohn-Gordon et al. (2018)'s original, fully incremental character-level generation approach produces captions that are not only more informative and globally more diverse, but also have lower quality, contain more repeated words as well as more out-of-vocabulary words. Generally, the mixed speaker that switches between fast and slow decoding is computationally more efficient and achieves a good balance in these various trade-offs maintaining, most notably, a better balance between quality and informativeness than a fully rational speaker.

2 Models

2.1 Image Captioning Model

We use Lu et al. (2017)'s adaptive attention model. The model's encoder uses a pre-trained CNN to represent images as feature vectors (we used ResNet152). A single LSTM layer with rectifier activation transforms the feature vectors into new vectors v^g. We concatenate the vector v^g with the word embedding vector w_t as the input for the decoder. Conditioned on image feature vector v_g and the hidden state vector h_{t-1} of the encoder from the previous time step, the attention module generates an attention map vector c_t. A single layer neural network transforms c_t and current hidden state vector h_t into a new vector, the final layer is a softmax over the vocabulary. While Lu et al. (2017) trained word level image captioning models, we trained a character-level model with the same architecture.

2.2 Pragmatic Reasoning

In the RSA model, a so-called rational speaker reasons about how an utterance would be understood by a listener, i.e., whether it allows the identification of the target. The speaker and listener are given a set of images W and one image $w^* \in W$ is known to the speaker as the target (see Figure 1). The rational speaker in RSA has an internal *literal speaker* who produces utterance candidates, i.e., a conditional distribution $S_0(u|w)$ which, in the simplest case, assigns equal probability to all true utterances $u \in U$ and zero probability to false utterances. A *pragmatic listener* L_0 then discriminates between

images given the utterance, as follows:

$$L_0(w|u) \propto \frac{S_0(u|w) * P(w)}{\sum_{w' \in W} S_0(u|w') * P(w')}$$

where $P(w)$ is a prior over possible target images. The pragmatic speaker S_1 is defined as:

$$S_1(u|w) \propto \frac{L_0(w|u)^\alpha * P(u)}{\sum_{u' \in U} L_0(w|u')^\alpha * P(u')}$$

where $P(u)$ is a uniform distribution over possible utterances U and $\alpha > 0$ is a rationality parameter determining the relative influence of the pragmatic listener in the rational speaker, see Cohn-Gordon et al. (2018) for further details. Essentially, the literal speaker S_0 corresponds to the standard formulation of the image captioning task, i.e., it generates descriptions for single target images. In contrast to this, the pragmatic speaker S_1 considers the respective context, i.e., the distractor images, during decoding.

We use Cohn-Gordon et al. (2018)'s character-incremental implementation of RSA that uses a character-level captioning model as the literal speaker S_0 and applies recursive pragmatic reasoning at each time-step, during unrolling a character-level neural speaker. While Cohn-Gordon et al. (2018) only reported results on decoding RSA with beam search, we compare greedy and beam search.

A Mixed Speaker While previous studies on RSA for neural generation have implemented fully rational speakers that apply pragmatic reasoning over the entire utterance or incrementally at each time-step of the decoding process, we propose a new scheme for decoding with RSA which we call a *mixed speaker*. This speaker is both literal and rational (or 'fast' and 'slow'), using different levels of reasoning during incremental decoding. We define our mixed speaker (S_x) to be: (i) rational (S_1) when generating the first character of a new word, i.e., at the beginning of the sequence or after generating a whitespace and (ii) literal (S_0) when generating other characters, i.e., not at the beginning of a word. This captures the intuition that the speaker can (and should) in many cases rely on its language model, e.g., when continuing words in a morphologically well-formed way. We test whether pragmatic reasoning at the beginning of words is enough to push the model towards being more informative, by giving more probability to initial letters of discriminative words. For instance, when the speaker describes a *big* and *yellow* object, pragmatic reasoning will be needed only at the beginning of the word, discriminating between *b* and *y*, depending on properties of the distractor objects.

3 Character-level Experiment

Our experiments compare three different speakers: the literal (or 'fast') speaker S_0, which simply decodes the language model, the rational (or 'slow') speaker S_1, which reasons at every time step, and the mixed (or 'fast and slow') speaker S_x.

3.1 Evaluation

Our evaluation setting is similar to Schüz et al. (2021), who investigated global diversity in pragmatic reasoning with word-level captioning models. Here, in addition, we analyze the repetitiveness of generated captions and evaluate informativeness in similar ways as in Cohn-Gordon et al. (2018), instead of using an external cross-modal retrieval model as in Schüz et al. (2021).

Data We performed experiments using the MSCOCO data set (Lin et al., 2014), with 82,783 and 40,504 images in the training and validation sets, respectively. Each image is annotated with around five captions. Following Cohn-Gordon et al. (2018), we train our speaker and listener models on distinct training splits. Because of this, we randomly split the training set into halves for model training. For evaluation, we randomly selected 1,000 images from the MSCOCO validation set.

Informativeness Following Cohn-Gordon et al. (2018), we train a listener model to predict target images for captions produced by a speaker. Given a set of potential target images and a generated caption, the listener ranks the images in terms of their likelihood. If the target image (i.e., the input of the speaker) is on top, the caption is accurate (reported as listener accuracy, L_{acc}, in Table 1). The clusters of potential target images were compiled based on caption similarity: For each target image, we select the two images as distractors whose annotated captions have the highest Jaccard similarity with the annotated captions of the target image.

Quality Evaluation We assess the quality of generated captions in terms of CIDEr scores (Vedantam et al., 2015), measuring the overlap with human captions. Since our model generates captions character by character, we report the absolute number of out-of-vocabulary types and tokens (OOV types and tokens) where we treat every

		α	Inform. L_{acc}	Quality CIDEr	Type Vocab IV	Type Vocab OOV	Token Vocab IV	Token Vocab OOV	Diversity TTR_{cap}	Diversity TTR_{cap_c}	Diversity TTR_{cap_2}	Diversity TTR_g
greedy	S_0	-	54.8	0.668	376	4	11273	5	0.760	0.870	0.916	0.166
	S_1	1	63.0	0.559	444	7	13136	7	0.714	0.822	0.884	0.177
	S_1	3	66.9	0.506	549	8	12990	8	0.720	0.819	0.881	0.201
	S_1	5	68.9	0.462	620	20	12846	25	0.723	0.817	0.884	0.212
	S_x	1	61.5	0.575	443	6	13127	8	0.718	0.825	0.888	0.175
	S_x	3	65.1	0.524	491	6	13014	6	0.723	0.820	0.888	0.189
	S_x	5	65.5	0.493	529	9	12998	10	0.728	0.824	0.890	0.198
beam	S_0	-	54.1	0.778	303	0	10369	0	0.841	0.930	0.964	0.160
	S_1	1	63.1	0.704	348	3	10359	3	0.826	0.915	0.952	0.171
	S_1	3	68.5	0.589	428	17	10360	32	0.796	0.872	0.927	0.193
	S_1	5	70.4	0.481	486	72	10730	199	0.769	0.839	0.902	0.209
	S_x	1	61.8	0.718	341	2	10497	2	0.828	0.918	0.952	0.170
	S_x	3	64.8	0.652	373	3	10580	3	0.812	0.899	0.939	0.180
	S_x	5	66.9	0.606	413	8	10694	8	0.797	0.884	0.927	0.190

Table 1: Evaluation of informativeness (listener accuracy), quality (CIDEr and OOV types and tokens), local diversity (TTR_{cap}, TTR_{cap_c}, TTR_{cap_2}) and global diversity (TTR_g) for literal (S_0), rational (S_1), mixed (S_x) speakers.

word token (occurring between whitespaces) that did not occur in the training set as an OOV token. In-vocabulary (IV) token and type counts are provided for comparison.

Diversity and Repetitiveness We measure diversity using different type-token ratios (TTR) (Youmans, 1990; van Miltenburg et al., 2018). TTR_g is calculated *globally* as the total number of types divided by the total number of tokens as in van Miltenburg et al. (2018) and Schüz et al. (2021). In contrast to this, TTR_{cap} is computed *locally* as the arithmetic mean of the TTR values for individual captions. While TTR_g reflects the general lexical diversity, TTR_{cap} is an indicator for word repetitions in captions. We supplement this with TTR_{cap_c} which is analog to TTR_{cap} but on captions filtered for stop words, i.e., indicating repetitions of content words. Finally, TTR_{cap_2} is based on bigrams and thus indicates the repetition of word combinations or phrases.

3.2 Informativeness–Quality Trade-Off

The results in Table 1 show that there is a systematic trade-off between informativeness and quality in character-level image captioning. All speakers that use some level of reasoning, S_1 or S_x with different α-values, achieve a higher listener accuracy but lower quality in terms of CIDEr than S_0. Beam search is generally beneficial for caption quality, according to the CIDEr scores shown in Table 1. However, it seems to interact with highly rational S_1 in unfortunate ways and leads to a drastic increase of the number of OOVs (see Figure 1 for an example). Here, RSA fails to achieve a good balance with its underlying language model. Our mixed speaker achieves a much better trade-off between quality and informativeness, especially in combination with beam search: $S_{x,\alpha=5}$ clearly outperforms $S_{1,\alpha=5}$ in CIDEr (more than 12 points improvement) and number of OOVs (only 8 types as compared to 72 for S_1), while its listener accuracy is only 3 points lower. From this, we conclude that occasional, word-initial pragmatic reasoning is highly effective and offers a decent balance between informativeness and quality.

Example 1 illustrates further, more general aspects of an informativeness–quality trade-off. S_0 and S_x generate more informative captions by integrating the background of the image as a discriminative feature (*"field"*). However, other features are also added, some of them inaccurate (e.g., *"grazing"*, *"fenced"*). This shows general limitations of this approach: Since discriminativity is the primary concern in RSA, other problems can arise, such as semantic inadequacies.

3.3 Local–Global Diversity Trade-Off

Results in Table 1 point to further trade-offs of pragmatic decoding, which generally lead to more repetitive captions being generated. In comparison to the literal speakers S_0, S_1 and S_x have lower TTR_{cap}, TTT_{cap_c} and TTR_{cap_2} scores. The mixed speaker attenuates this effect, but has still lower local TTR as the fully literal speaker. This should not happen in theory, since it is questionable whether repeating is a useful strategy for making utterances more discriminative. It could even be seen as a strategy that violates the Gricean maxim of relevance (Grice,

1975) – see the caption of S_x in Figure 1 for a representative example. Interestingly, however, increases in local repetitiveness are combined with increases in global diversity. Thus, speakers which are more repetitive also use a larger vocabulary (S_1 and S_x use more absolute types and have higher TTR_g). RSA counteracts the tendency of beam search to globally reduce the vocabulary, but, here, greedily decoded speakers achieve higher numbers of types and TTR_g. This indicates that RSA might be a useful approach to generating, e.g., less frequent words when the communicative context makes this necessary, as previously suggested in Schüz et al. (2021).

4 Conclusion

In this paper we have replicated Cohn-Gordon et al. (2018)'s approach to character-level pragmatic image captioning and have evaluated it not only with respect to informativeness, but also quality, repetitiveness, and global diversity of the model outputs. Our results show various trade-offs in character-level captioning: models that are more informative and rational produce captions that are of lower quality and contain more out-of-vocabulary words. In terms of diversity, we find that character-level RSA *increases* the amount of word repetitions within captions. Interestingly, at the same time, it also increases global diversity and leads to a larger vocabulary being exploited by the underlying language model. This analysis fully confirms and extends findings on word-level decoding methods facing different types of trade-offs as a result of additional objectives introduced into the generation process at the decoding stage (e.g., Li et al., 2016; Vijayakumar et al., 2018; Ippolito et al., 2019; Schüz et al., 2021).

Our analysis also shows that these trade-offs can be countered by our mixed, 'fast' and 'slow', speaker that applies reasoning in a simple, word-initial fashion. Future work could explore further ways of controlling *when* reasoning is needed during decoding and generalize this to word-level decoding. As character-level image captioning is arguably not state-of-the-art, some of the effects that we report in this case study might not generalize to more powerful – especially word-level – models. Nevertheless, we believe that the trade-offs observed in this pilot study could be explored in other task that require the generation of long sequences (e.g., image paragraphs, longer responses in an interaction) and that the effectiveness of mixed pragmatic decoding might be an interesting avenue for such tasks.

References

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Austin, Texas. Association for Computational Linguistics.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press, Cambridge, UK.

Reuben Cohn-Gordon, Noah Goodman, and Christopher Potts. 2018. Pragmatically informative image captioning with character-level inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 439–443.

Reuben Cohn-Gordon, Noah Goodman, and Christopher Potts. 2019. An incremental iterated response model of pragmatics. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 81–90.

Judith Degen and Michael K Tanenhaus. 2016. Availability of alternatives and the processing of scalar implicatures: A visual world eye-tracking study. *Cognitive science*, 40(1):172–201.

Judith Degen and Michael K Tanenhaus. 2019. Constraint-based pragmatic processing. *Handbook of Experimental Semantics and Pragmatics*.

Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018. Unified pragmatic models for generating and following instructions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1951–1963, New Orleans, Louisiana. Association for Computational Linguistics.

Alexia Galati and Susan E. Brennan. 2010. Attenuating information in spoken communication: For the speaker, or for the addressee? *Journal of Memory and Language*, 62:35–51.

H. P. Grice. 1975. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41–58. Academic Press, New York.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Daphne Ippolito, Reno Kriz, João Sedoc, Maria Kustikova, and Chris Callison-Burch. 2019. Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3752–3762, Florence, Italy. Association for Computational Linguistics.

Daniel Kahneman. 2011. *Thinking, Fast and Slow*. Macmillan.

Boaz Keysar, Dale J. Barr, and William S. Horton. 1998. The egocentric basis of language use: Insights from a processing approach. *Current Directions in Psychological Science*, 7:46–49.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 375–383.

Emiel van Miltenburg, Desmond Elliott, and Piek Vossen. 2018. Measuring the diversity of automatic image descriptions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1730–1741, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ehud Reiter. 1991. A new model of lexical choice for nouns. *Computational Intelligence*, 7(4):240–251.

Walter Schneider and Richard M Shiffrin. 1977. Controlled and automatic human information processing: I. detection, search, and attention. *Psychological review*, 84(1):1.

Simeon Schüz, Ting Han, and Sina Zarrieß. 2021. Diversity as a by-product: Goal-oriented language generation leads to linguistic variation. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 411–422, Singapore and Online. Association for Computational Linguistics.

John Searle. 1969. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4060–4067, Minneapolis, Minnesota. Association for Computational Linguistics.

Yik-Cheung Tam. 2020. Cluster-based beam search for pointer-generator chatbot grounded by knowledge. *Computer Speech & Language*, 64:101094.

Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. 2017. Context-aware captions from context-agnostic supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 251–260.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Julia White, Jesse Mu, and Noah Goodman. 2020. Learning to refer informatively by amortizing pragmatic reasoning. In *Proceedings of the 42nd Annual Meeting of the Cognitive Science Society*.

Gilbert Youmans. 1990. Measuring lexical style and competence: The type-token vocabulary curve. *Style*, 24:584–599.

GraphPlan: Story Generation by Planning with Event Graph

Hong Chen[1,3], **Raphael Shu**[1], **Hiroya Takamura**[2,3], **Hideki Nakayama**[1,3]
The University of Tokyo[1], Tokyo Institute of Technology[2]
National Institute of Advanced Industrial Science and Technology, Japan[3]
{chen, nakayama}@nlab.ci.i.u-tokyo.ac.jp
shu@deeplearn.org, takamura.hiroya@aist.go.jp

Abstract

Story generation is a task that aims to automatically generate a meaningful story. This task is challenging because it requires high-level understanding of the semantic meaning of sentences and causality of story events. Naive sequence-to-sequence models generally fail to acquire such knowledge, as it is difficult to guarantee logical correctness in a text generation model without strategic planning. In this study, we focus on planning a sequence of events assisted by event graphs and use the events to guide the generator. Rather than using a sequence-to-sequence model to output a sequence, as in some existing works, we propose to generate an event sequence by walking on an event graph. The event graphs are built automatically based on the corpus. To evaluate the proposed approach, we incorporate human participation, both in event planning and story generation. Based on the large-scale human annotation results, our proposed approach has been shown to provide more logically correct event sequences and stories compared with previous approaches.

1 Introduction

Narrative intelligence (Mateas and Sengers, 2003) is a form of humanistic artificial intelligence that requires the system to organize, comprehend, and reason about narratives, and then, produce meaningful responses. Story generation tasks can be considered as a test bed for examining whether a system develops a good understanding of the narratives. In addition to leaving the model to output random sequences, the model is usually given a specific topic (e.g., title or prompt) or visual information (e.g., image or video). One straightforward approach for these story generation tasks is to leverage a sequence-to-sequence model to predict sentences sequentially. Although the model can be trained to capture the word-prediction distribution from the training data, it has two serious drawbacks when applied to generating stories: 1) A conditional language model (i.e., the decoder) tends to assign high probabilities to generic, repetitive words, especially when beam search is applied in the decoding phase (Holtzman et al., 2019); and 2) sequence-to-sequence models often fail to produce logically correct stories.

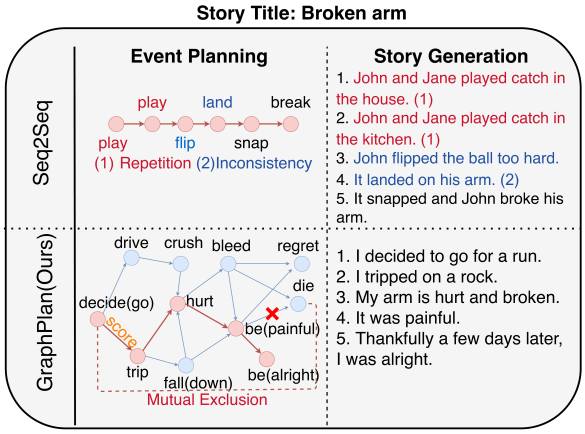


Figure 1: Comparison between sequence-to-sequence model and GraphPlan (ours). Two problems occur in the sequence-to-sequence model when generating events: repetition and logical inconsistency. Repeated words (e.g., play) in the storyline result in repeated sentences in the generated stories. In addition, the logic between "land" and "snap" lacks causality, thus generating incoherent stories. On the contrary, our GraphPlan method does not rely on any language model, and applies beam search on the event graph based on a well-designed score function. The mutually exclusive set further ensures global logical consistency for the planned events.

Recently, there has been significant interest in decomposing story generation into two phases: Planning and generation (Yao et al., 2019; Goldfarb-Tarrant et al., 2019; Xu et al., 2018; Fan et al., 2019). Planning (Meehan, 1976; Riedl and Young, 2010) creates a high-level abstraction or a blueprint

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 377–386,
Aberdeen, Scotland, UK, 20-24 September 2021.

that encourages the generator to focus on the flow of a story, similar to making an outline before writing. The planned elements are referred to as *events* in several papers. However, the detailed definition of events varies. For instance, an event can be represented as a verb argument pair (e.g., *(admits, subj)*) (Chambers and Jurafsky, 2008): tuple of subject, verb, object and modifier or "wildcard" (e.g., *(PERSON0, correspond-36.1, empty, PERSON1)*) (Martin et al., 2018; Ammanabrolu et al., 2020) or reconstructed verb phrase (e.g., *decide(go)*) (Peng and Roth, 2016). In this paper, we follow Peng and Roth (2016) to represent an event with verb phrases.

Existing approaches (Goldfarb-Tarrant et al., 2019; Martin et al., 2018; Ammanabrolu et al., 2020) regard event generation as an abstracted case of story generation. In other words, they treat each event as one token and use a sequence-to-sequence model to plan the events. Our preliminary experiments show that repetition and logical inconsistency problems occur in the event sequence; further, the same problems occur in the generated stories. Figure 1 shows an example using sequence-to-sequence event planning. Both events and stories are repeated and illogical.

In this study, instead of a sequence-to-sequence model for event planning, we propose a planning method, **GraphPlan**. To plan the event, GraphPlan walks on a topic-specific event graph with a beam search. Event graphs have been adopted for story generation even before the emergence of neural-based models (Weyhrauch, 1997; Chen et al., 2009; Regneri et al., 2010; McIntyre and Lapata, 2010; Li et al., 2013). An event graph represents the logical flow of events based on the facts presented in a corpus. We can walk on a learned event graph and produce a reasonable event sequence. We follow the graph setting in Li et al. (2013), wherein each graph is composed of event nodes, functions and a set of mutually exclusive events.

To generate a story, we first identify the topic based on the input (e.g., title or image), and subsequently, retrieve a related event graph. We then plan the events by running a beam search with a score function that considers event-event coherence and input-event coherence into account. Finally, a *story generation* module transforms the planned event sequence into a readable story. Figure 2 depicts the entire pipeline of our proposed approach.

We conducted experiments on open story generation to evaluate how event graphs benefit tasks. Our approach significantly outperforms baseline models that generate events with sequence-to-sequence models in terms of logical consistency. We also conducted Story Cloze Test (SCT) to further validate the effectiveness of the event graphs and the mutually exclusive sets. Our contributions can be summarized as follows:

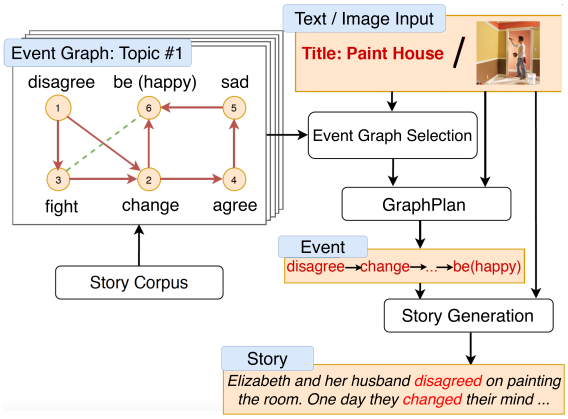


Figure 2: Overview of our approach. In the preprocessing step, we cluster the stories into K topics and build an event graph for each topic. In the planning step, an event graph selection module selects an event graph based on the input. Then, a related event graph is retrieved. The event planning model generates a sequence of events. Finally, based on the input and planned events, a story generation module generates the story. The dashed line denotes mutually exclusive events that are difficult to coexist in the same storyline.

- We propose a score-based beam search approach to plan story events with an event graph.
- Compared with baseline models, our graph-based planning approach results in much better logical correctness in story generation tasks according to human evaluation.
- Experiments on SCT directly confirm the high accuracy of the proposed event planning approach.

2 Related Work

Planning for Story Generation Several approaches have been explored to plan the skeleton of a story before its generation. Before the emergence of neural-based models, Reiter and Dale (1997) and Riedl (2010) attempted to use hand crafted rules to arrange actions into character sequences. Recently, with the help of neural sequence-to-sequence models, Xu et al. (2018) proposed to generate multiple key phrases and expand them into a complete story. A built-in key phrases generation module is used in their model architecture. In contrast to Xu et al.

(2018), some works have explicitly plan a sequence of events (Martin et al., 2018; Ammanabrolu et al., 2020; Tambwekar et al., 2019; Fan et al., 2018; Rashkin et al., 2020; Ammanabrolu et al., 2021), keywords (Yao et al., 2019; Ippolito et al., 2019; Goldfarb-Tarrant et al., 2020) or actions (Fan et al., 2019) before generating the story based on the planned items.

All of these planning models rely on a language model for planning, without following an external structure of events, which results in degraded performance (Holtzman et al., 2019). Compared with these works, the main contribution of this paper is to propose a planning method based on automatically created event graphs. Instead of a language model, we use score-based beam search to generate a sequence of events by walking on the graph.

Graph-based Story Planning An event graph is a variant of a plot graph whose nodes represent events. Previous research has made progress on generating stories from plot graphs (Weyhrauch, 1997; Chen et al., 2009; Regneri et al., 2010; McIntyre and Lapata, 2010; Li et al., 2019). Li et al. (2013) proposed a plot graph on story generation tasks, which is relevant to our work. They crowdsourced the story corpus and manually created plot nodes and edges in the graph. In their graph, mutually exclusive events are not allowed to be present in the same story.

In this work, both the event graphs and mutually exclusive sets are automatically generated. We further propose an event planning method that considers the relations between events and various inputs (i.e., title or image).

3 Event Graph Construction

As a preprocessing step, we first extract events automatically from a corpus. Then, we divide the corpus into several topics. Finally, we build an event graph for each topic.

Data-based Event Extraction Following Peng and Roth (2016), we represent each event with a verb phrase. Unlike other representations, a verb phrase is the minimum unit in one sentence that is abstract, simple, and comprehensible for humans. From our observation, this representation does not have a severe sparseness problem. In statistics, each event in the graph can link to over three next possible events on average. Note that our work does not investigate event representation; instead, we focus on planning a more logical event sequence. Specifically, as a preprocessing step, we parse all sentences with semantic role labeling [1] and extract the verb phrases. If an extracted verb has an argument with the semantic role "AM-NEG" (negation) for a verb, we add *(not)* before it (e.g., *(not)take*). If a verb is followed by a preposition, we append the prepositional word to the verb (e.g., *take(over)*). If the label is "AM-PRD" (secondary predicate), we make an event from it (e.g., *be(excite)*). Finally, if two verbs are close to each other within five-word distance in the corpus, we combine them to make an event (e.g., *decide(buy)*). All words in the event are stemmed using NLTK (Bird et al., 2009).

Topic Modelling Generally, a story dataset contains various topics, ranging from animals to health to robbery. Here, we use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to infer the topics in the corpus. Considering that the relation between events drastically changes according to the topic, in this study, we build an independent event graph for each topic. Formally, we denote $e_1^k, \ldots, e_t^k$ the event set from the stories that belong to the k-th topic $\mathcal{T}^k$ in the corpus. These events would be used as nodes for the event graph of $\mathcal{T}^k$. LDA clusters the stories and thus reduces the amount of unique events in each graph, which will make the graph walking algorithm more efficient.

Event Connection After collecting events from a corpus for each topic, we need to determine connections among these events to build a graph. The connections are represented as directed edges whose direction indicates possible next events. In practice, if events e_i and e_j occur adjacently in the text, we add an edge $e_i \rightarrow e_j$. An example of an event graph is presented in Figure 2.

Mutually Exclusive Set Following the graph setting in Li et al. (2013), there are events (e.g. "die" and "be(happy)") that are mutually exclusive, and thus, cannot be placed in the same story. These mutually exclusive relations are considered as exceptions, which are difficult to represent along with the event graph. We create a held-out set consisting of mutually exclusive event pairs for each graph.

To identify these mutually exclusive events from the constructed graphs, we prepare an event-event

[1] https://demo.allennlp.org/semantic-role-labeling/semantic-role-labeling

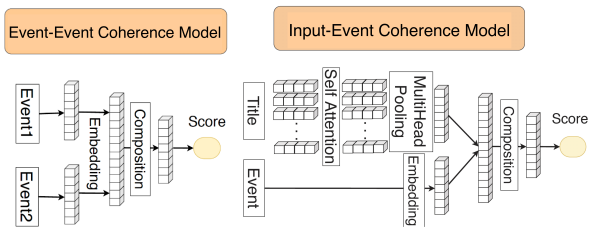


Figure 3: Coherence models were used in this study. The event-event coherence model outputs a coherence score for two events. The input-event coherence model takes a title and an event as input. Both coherence models finally produce a score within 0 to 1. These coherence scores determine the next event when running beam search.

coherence model to detect the coherence score between two events. Consequently, we prevent two events with low coherence scores from coexisting in the planned events. The model architecture is based on compositional neural networks (Granroth-Wilding and Clark, 2016), as shown in Figure 3. The model takes two events (e_i, e_j) represented with unique embeddings, and outputs a coherence score normalized with the sigmoid function $f_{event}(e_i, e_j) \in [0, 1]$. We use contrastive training to optimize the model. Here, positive examples are the events extracted from the same story or title, whereas negative samples are randomly sampled from the events in different stories. Let (e_i, e_j) denote a positive pair of events, and $\tilde{e}_j$ denotes a randomly sampled event. The training loss for the event-event coherence model is defined as

$$L_{event} = \max(0, -f_{event}(e_1, e_2) + f_{event}(e_1, \tilde{e}_2) + m), \quad (1)$$

where m is a fixed margin. Finally, we consider two events as mutually exclusive if the coherence score falls below a certain threshold τ.

On average, after considering the mutually exclusive sets, each event graph can still plan over one million different possible event sequences. Please refer to the supplementary materials for more statistical details on event graphs. Additionally, these in-topic event graphs can be hierarchically combined into a larger graph, if the model is required to generate longer discourse-level stories. This will be a future direction for our work.

4 GraphPlan: Planned Story Generation using Event Graph

In this section, we describe our approach for planned story generation. We separated the entire pipeline into two steps: 1) Our GraphPlan walks on the event graph and produces a sequence of events as a blueprint of the story. 2) The story generation module then finalizes the text following the planned events.

4.1 GraphPlan

Each topic has had a corresponding event graph. Before story generation, we propose GraphPlan to plan event sequences from the event graph. These planned events will be used to guide the story generation module in the next step. GraphPlan comprises two steps. 1) Selecting an event graph for the input (i.e. title or image); and 2) generating an event sequence by walking on the graph.

Event Graph Selection First, we identify the topic of inputs to retrieve the corresponding event graph. Depending on the tasks, the inputs can be titles for open story generation, or images for the visual storytelling task. If the input is a piece of text, we directly use the LDA model, trained earlier, to identify the topic.

Event Sequence Generation Once we identify the topic of input, we walk on the corresponding event graph to generate a sequence of events. In our experiments, we found that an autoregressive language model tends to produce repetitions, resulting in degraded performance. Hence, we propose to use a score-based generation method. The algorithm can be seen as a type of beam search, in which the candidate event sequences are ranked by a score function. Starting from a random event e_1, we progressively search for the next event e_t in the following candidate set:

$$\{e_t \mid e_t \in \text{Graph}(e_{t-1}), \\ e_t \notin \text{Exclusive}(e_1, \ldots, e_{t-1})\}, \quad (2)$$

where $\text{Graph}(\cdot)$ returns a set of possible next events in the graph, and $\text{Exclusive}(\cdot)$ returns a set of mutually exclusive events. This filtering step significantly reduces the number of candidate events.

To select the event from the candidate set, we rank all remaining candidate events with the following score function:

$$\text{Score}(e_t) = \frac{1}{\sum_{i=0}^{t-1} \lambda^i} \sum_{i=0}^{t-1} \lambda^i \log f_{\text{event}}(e_i, e_t) \\ + \log f_{\text{input}}(x, e_t) \quad (3)$$

where the first term of the score function sums the event-event coherence score of candidate event e_t to each partially generated event e_i and gives

more weight to recent events. λ denotes the decay rate. Then, a decayed average is applied over the score. The model used in producing the event-event coherence is the same model that is used to detect mutually exclusive events. The second term is an input-event coherence score $f_{\text{input}}(x, e_t)$, which indicates the coherence score between event e_t and the input x. We propose an input-event coherence model to compute this score; refer to Figure 3 for details about parameterization. For the task of open story generation, the input-event coherence model is implemented via compositional neural networks, where input x in Equation (3) is the title. As common practice for beam search, we set a budget for the number of candidates to explore (i.e., beam size). The candidates with the highest scores are maintained in the beam. The final candidate with the highest score is selected as the result.

4.2 Story Generation Module

The generated event sequence will be sent to a story generation module, which converts the events into a story; this module can be any type of model. Recently, large pre-trained language models show remarkable capacity for generating knowledgeable and informative sentences. Utilizing these advantages, the planned events are more likely to be logically connected in the generated story. Therefore, we apply GPT2-small (Radford et al., 2019) as our story generation module. During the training, we feed the module with the title words and events. A special token "<EOT>" separates the title and events and another special token "<SEP>" is placed in every interval of the events. "<|endofinput|>" token is added at the end of the input. In addition, we train an RNN-based sequence-to-sequence model that is fed with the same inputs for comparison.

However, as stated in Yao et al. (2019); Tan et al. (2020), the exposure bias problem occurs when plan-write strategy is applied. To mitigate this problem, we alternatively add two types of noises into the inputs: 1) Mask 20% events with a "[MASK]" token; and 2) mask all events. The first noise encourages the model to generate sentences, referring to all planned events, while, the second noise promotes the model to generate more stories related to the title. The effectiveness of two noises are analyzed in the supplementary material.

5 Experiment

5.1 Experiment Settings

We design two experiments to explicitly evaluate event and story quality. First, we calculate the diversity score and conduct human evaluation on the planned events. Secondly, we use the story generation module to transform the events into full stories and conduct human evaluation to evaluate the story quality. Moreover, to further verify the correctness of our GraphPlan, we conduct experiments on Story Cloze Test. The details of model implementations for all experiments can be found in the supplementary material.

5.2 Dataset

ROCStories Corpora (Mostafazadeh et al., 2017) consists of 98,162 stories with titles that we use as the input and a five-sentence story that we use as the target. We chose this dataset as a testbed since the sentences included inside are simple, making it easier to accurately capture the events. We split these stories into 8:1:1 for training, validation and testing. We applied clustering to the training split (i.e., 8 of 8:1:1) and obtained 500 topics, in which each topic represents one specific domain. Each story is generated from one specific domain in the following experiments. Gold event sequences that are used in planning methods are extracted from the stories in the corpus.

5.3 Baseline

S2S Following Yao et al. (2019), we use a sequence-to-sequence model (Bahdanau et al., 2015), which straightforwardly generates events by the titles.
S2S(R) To build a more competitive baseline, we adopt reward shaping in the sequence-to-sequence model. As in (Tambwekar et al.), we apply a policy gradient on

$$\begin{aligned} \nabla_\theta J(\theta) &= R\left(e_i\right) \nabla_\theta \log P\left(e_i \mid e_1, \ldots, e_{i-1}; \theta\right) \\ R(v) &= \alpha \times r_1(v) \times r_2(v) \\ r_1(v) &= \log f_{\text{input}}(x, v) \\ r_2(v) &= \frac{\sum_{e \in E \wedge e \neq v} \log f_{\text{event}}(e, v)}{N-1} \end{aligned} \tag{4}$$

where e denotes the set of the events in the planning sequence; E denotes the events in the story; N denotes the number of events in the story; x denotes the input title and α denotes the normalization constant across the events in each training sample. During training, the gradient from e_i is multiplied by the reward $R(e_i)$, which is propor-

Diversity	S2S	S2S(R)	GP	GOLD
Dist-1	10.17%	11.35%	**20.54%**	24.92%
Dist-2	56.55%	58.92%	**78.12%**	87.75%

Table 1: Diversity of planned events. Both sequence-to-sequence models achieve low diversity, while Graph-Plan can achieve high diversity.

tional to $r_1(e_i)$ and $r_2(e_i)$. In brief, $r_1(e_i)$ increases when e_i is more related to the input x, while $r_2(e_i)$ become larger when e_i is more likely to coexist with all events $\{e|e \in E \wedge e \neq e_i\}$. This method forces the model to focus on the event that has a high coherence score with the input and events in each training sample.

GR In this method, we apply random walk on the event graphs while considering mutually exclusive sets. We aim to show the importance of using coherence models by comparing it with this method.

GP(Ours) This is our proposed method that plans events on an event graph via mutually exclusive set and coherence models.

5.4 Event Quality

We plan the events on 1000 randomly selected test data with different baselines models and our proposed model. We first tested the diversity of the generated sequences and calculate Distinct-1 and Distinct-2 scores to measure the diversity for all generated events. Table 1 shows that the sequence-to-sequence model suffers from producing repeated unigram and bigram events. Graph plan produces more events in the full event set (more unigrams) and produces more combinations between events (more bigrams).

To further evaluate the quality of planned events, we conduct human evaluation. Instead of using overly abstract event representation, as in (Tambwekar et al., 2019), we use the verb phrase, which is more easily understood by humans. Thus, we request the annotators to compare the event sequences using two criteria: Relevance and logicality. Table 2 shows the human evaluation results. These results show that our planned events (i.e., verb phrases) are more related to the input title and can be easily transformed into a story.

Table 3 shows some of the examples generated by different methods. The results show that sequence-to-sequence models tend to generate repetitive events. Specifically, they tend to output the events that occur with high frequency in the corpus, such as "be"(there is sth.) and "go" (sb. go somewhere). This is common for a model

Choices(%)	GP vs S2S		GP vs S2S(R)		GP vs GR	
	GP	S2S	GP	S2S(R)	GP	GR
Relevance	**47.0**	17.8	**52.0**	26.0	**56.3**	12.9
Logical	**53.5**	14.9	**55.2**	26.0	**51.5**	17.8

Table 2: Human evaluation of event planning. Cohen's Kappa coefficient (κ) for all annotations is in moderate agreement (0.4-0.6). Sign tests show that p-values are < 0.01 for all pairwise comparisons.

Title	Married too fast
S2S	be→be→be→fall→marry
S2S(R)	be→go(up)→ask→say→marry
GR	want(do)→go→sit→wonder→call
GP	feel→decide→begin→start→regret
Title	New glasses
S2S	sit→be(unhappy)→have→go→find
S2S(R)	break→need→go→get→be(glad)
GR	wake(up)→be→(not)care→take→ make
GP	buy→wear→break→shatter→decide(buy)
Title	Grilled cheese
S2S	love→be→decide→forget→end(up)
S2S(R)	make→get→go→go→look→see
GR	feel(comfortable)→like→smile→decide→feel(full)
GP	melt→put→decide(roast)→burn→taste

Table 3: Examples of the planned events.

trained under the framework of the maximum likelihood estimation method. Although reward shaping (S2S(R)) helps with this problem substantially, it is still not eliminated. Without the limitation of the coherence score, GR walks on graphs randomly to produce a sequence. As the graph is not small, achieving a good event sequence is extremely challenging. Our proposed GP produces more logical and diverse events, using which humans can easily tell a story based on these given events.

5.5 Open Story Generation

The goal of event planning is to generate more related and logical coherent stories. Human evaluation of the event sequence is subjective and tricky since the event is highly abstract. To prove that better event planning improves the story quality, we generate stories using the planned events and conduct human evaluation to assess the relevance, logicality, interestingness and overall scores. We use the story generation module (i.e. GPT2 and RNN) to transform these planned events into the full stories. We compare the following methods:

GPT2. This is a large scale language model that has shown remarkable performance in generating stories in recent research. In this method, we directly input the title to GPT2 and generate full stories.

***+GPT2**. We associate the event planning meth-

Choices(%)	GP vs GPT2		GP vs S2S		GP vs S2S(R)		GP vs GR		GP vs GP+RNN	
	GP	GPT2	GP	S2S	GP	S2S(R)	GP	GR	GP	GP+RNN
Relevance	33.3	**41.6**	**38.4**	28.3	**40.8**	20.3	**67.8****	23.2	**37.2***	23.3
Interestingness	**47.9**	41.6	**40.1**	30.8	**43.2**	34.0	**60.7***	39.3	**44.2****	24.5
Logicality	**64.6****	27.1	**42.6****	19.5	**44.6***	25.0	**62.5****	33.9	**37.2**	32.6
Overall	**56.3***	35.4	**42.6****	21.6	**42.3***	24.6	**64.3****	30.4	**46.5**	37.2

Table 4: Human evaluation of open story generation. (+GPT2) are omitted for all methods, except for GP+RNN. We calculate Cohen's Kappa coefficient (κ). All the results are in moderate agreement (0.4-0.6). We conduct sign tests to show the significant difference. (*) and (**) denotes p-value is < 0.05 and < 0.01 respectively. The results show that our method achieves significant improvement on logicality, thereby getting better overall performance.

	Title: Fire next door
GPT-2	*After the fire, the fire was out. The family who lived next door started to panic, so they followed it. [The firehouse across the street was still burning. The fire house was still burning next door.] (Repetition)*
S2S	make → think → go → see → get → apologize
	*Lisa's new neighbor **made** a big fire outside his house. Lisa **thought** she would be sorry. She **went** outside to check on him. She **saw** that he was not there. Lisa's neighbor could not **get** a word with him and **apologized**.*
S2S(R)	try → tell → leave → throw → take
	*Sally **tried** to fight a burning bush. The homeowner **told** her she couldn't do it. She **left** a piece of wood in the bush next door. A woman **threw** the wood at sally. Sally **took** the wood to the fire department.*
GR	quit → be → look(buy) → find → go
	*I **quit** my job when i was 16. The only thing I ever didn't do **was** take care of my children. Now I'm **looking to buy** a car. I **found** a cheap car on ebay. Now i can **go** out in the sun every day.*
GP+RNN	catch(on) → come → go(off) → evacuate → extinguish
	*The fire alarm was **caught on**. The firefighters **came** to help. The fire alarm **went off**. Everyone **evacuated**. The firemen **extinguished** it before it could go off.*
GP	catch(on) → come → go(off) → evacuate → extinguish
	*The house next door **caught on** fire. The fire department **came** to the scene. The fire alarm **went off**. The entire neighborhood **evacuated**. The fire department **extinguished** the fire.*
Gold	*John woke up smelling like something was burning. He went outside. He saw the fire next door. He called the authorities. The firemen came to put out the fire.*

Table 5: Examples of open story generation. Red words represent the events.

ods with GPT2, which is used in the story generation module. Thus, we compare S2S+GPT2, S2S(R)+GPT2, GR+GPT2, and GP+GPT2.

GP+RNN. In this method, we use an RNN-based sequence-to-sequence model to generate the full story, which takes the title and events as inputs. We compare this method to GP+GPT2 to show the effectiveness of large scale language models in transforming the events into the stories.

We conducted human evaluation on Amazon Mechanical Turk (AMT) over four aspects: *relevance* (whether the story is related to the topic), *interestingness* (whether the story content and style are interesting), *logicality* (whether the story is logical), and *overall* (overall quality). The full details of the human evaluation are listed in the supplementary materials. We randomly sampled 300 titles from the testing set and generated stories using each method. Pairwise comparison was conducted for each criterion and each sample was assigned to two different workers to avoid randomness or personal bias. Table 4 shows that our approach performs better with respect to logicality and overall. In particular, our method greatly outperforms other planning methods in the logicality measure, which suggests that our planned events are logically sound. We believe that the following two factors are the primary reasons for improved logic: 1) Each event graph is built from the corpus; thus, walking on the graph retains the events' logical relations; and 2) the coherence models filter the candidates, and the mutually exclusive set further eliminates the non-logical combinations when planning the events. Table 5 shows examples of stories generated using these methods. We show both the planned events and stories. Directly using GPT2 produces repeated sentences. Both equipped with an auto-regressive model for event planning, the events planned by S2S and S2S(R) fail to output satisfactory results, leading to a low logicality score in the generated sentences. Because there is no restriction on the event selection in GR, the event produced could be irrelevant to the title and even mutually contradictory. Using our proposed method, GP can plan a reasonable set of events, and thus, generate the most logical story.

5.6 Story Cloze Test (SCT)

To better validate the effectiveness of our event graphs and the mutually exclusive relation between events, we conducted SCT. This task aims to select the correct ending sentence from two candidates. We incorporated the event features generated by

ACC(%)	Test v1.0	Test v1.5
DiffNet	77.60	64.45
DiffNet+Origin	78.87	67.64
DiffNet+RandomWalk	79.36	68.09
DiffNet+GraphPlan	**80.15**	**69.45**

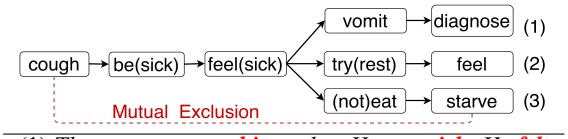
Table 6: Results on story cloze test. From the results, events planned by our event graphs and mutually exclusive sets have positive effects on this task.

different methods into the SCT. The accuracy of SCT reflects the quality of the event features. The event feature is learned by a mask language model (MLM) (i.e., the BERT model with fewer parameters) (Devlin et al., 2019). If the training event sequence is more logical and reasonable, the feature learned by MLM would better fit the SCT. To prove that our event graph and mutually exclusive relation can help us to generate reasonable event sequences, we compared the features generated by the MLM model with different training data: (1) **Origin**: Event sequences extracted from ROCStories Corpora. (2) **RandomWalk**: Random walk on the event graphs and sample training data. (3) **GraphPlan**: Using our planning method to generate training data. Note that the input-event coherence score is excluded in the score function because no input being given. We then use the event feature and adopt the state-of-the-art model, DiffNet (Cui et al., 2020) for SCT. For fair comparison, RandomWalk and GraphPlan included the same number of event chains in the dataset during training. Further details of the model can be found in the supplementary material.

Table 6 presents the results of SCT. This shows that RandomWalk and GraphPlan achieve better scores in both SCTv1.0 and v1.5, proving that our event graphs and mutually exclusive events have positive effects on event planning.

6 Discussion

As mentioned before, the stories can be easily controlled by modifying the events. Table 7 shows an example. Selecting different upcoming events for "feel(sick)" will change the following storyline. Moreover, our experiment shows that event graphs can produce more logical stories than planning via language models. Here, we give an empirical explanation. Sequence-to-sequence models usually fail to capture long-term relations and order information in the event sequence. The decoder is not guaranteed to account for all previous events during



(1) *The man was **coughing** a lot. He **was sick**. He **felt sick** for days. He **vomited** on the couch. He **was** later **diagnosed** with the flu.*
(2) *The man was **coughing** a lot. He **was sick**. He **felt sick**. He **tried** to **rest** for an hour. The man **felt** better !*
(3) *The man was **coughing** a lot. He **was sick**. He **felt sick**. He **couldn't eat** anything. He **starved** himself.*

Table 7: Example of controllable generation. (1) and (2) extends different events after "feel sick" to achieve different endings and (3) shows logical inconsistency when generating with two mutually exclusive events

decoding. At this point, our approach applies event-event coherence scores, which forces the model to consider long-term relations during planning. In addition, the order of events is captured from the gold cases, which can be guaranteed in our event graphs. Moreover, mutually exclusive sets help us to determine whether two events can co-occur in one sequence regardless of the distance between two events. Table 7 provides an example. Note that "cough" and "starve" are considered mutually exclusive events in our event graph. If we generate a story based on this event chain, the last sentence "he starved himself" is unreasonable in this case.

7 Conclusion

In this study, we demonstrate that a graph-based event planning approach can indeed produce more natural event sequences compared with conventional language models. We propose to walk on automatically learned event graphs by performing beam search using a score function dedicated for event planning. Then, the story is generated follows the planned events.

We evaluate our approach on event planning and open story generation with large-scale human judgements. The results show that our proposed approach clearly outperforms the non-planning baseline and the sequence-to-sequence model-based planning models. In human evaluation, the events and the stories generated by our proposed method are believed to be more logical and coherent. An additional experiment on Story Cloze Test further proves the advantages of event graphs and mutually exclusive sets.

8 Acknowledgements

We thank the anonymous reviewers for the useful comments. This work was supported by JSPS KAKENHI Grant Numbers JP19H04166 and based on results obtained from a project JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). For experiments, computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by National Institute of Advanced Industrial Science and Technology (AIST) was used.

References

Prithviraj Ammanabrolu, Wesley Cheung, William Broniec, and Mark O Riedl. 2021. Automated storytelling via causal, commonsense plot ordering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5859–5867.

Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung, Zhaochen Luo, William Ma, Lara J Martin, and Mark O Riedl. 2020. Story realization: Expanding plot events into sentences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7375–7382.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797.

Sherol Chen, Mark J Nelson, Anne Sullivan, and Michael Mateas. 2009. Evaluating the authorial leverage of drama management. In *AAAI Spring Symposium: Intelligent Narrative Technologies II*, pages 20–23.

Yiming Cui, Wanxiang Che, Wei-Nan Zhang, Ting Liu, Shijin Wang, and Guoping Hu. 2020. Discriminative sentence modeling for story ending prediction. In *AAAI*.

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

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660.

Seraphina Goldfarb-Tarrant, Tuhin Chakrabarty, Ralph Weischedel, and Nanyun Peng. 2020. Content planning for neural story generation with aristotelian rescoring. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4319–4338.

Seraphina Goldfarb-Tarrant, Haining Feng, and Nanyun Peng. 2019. Plan, write, and revise: an interactive system for open-domain story generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 89–97.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2727–2733. AAAI Press.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Daphne Ippolito, David Grangier, Chris Callison-Burch, and Douglas Eck. 2019. Unsupervised hierarchical story infilling. In *Proceedings of the First Workshop on Narrative Understanding*, pages 37–43, Minneapolis, Minnesota. Association for Computational Linguistics.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. 2013. Story generation with crowdsourced plot graphs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Wei Li, Jingjing Xu, Yancheng He, ShengLi Yan, Yunfang Wu, and Xu Sun. 2019. Coherent comments generation for Chinese articles with a graph-to-sequence model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4843–4852, Florence, Italy. Association for Computational Linguistics.

Lara J Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Michael Mateas and Phoebe Sengers. 2003. *Narrative intelligence*. J. Benjamins Pub.

Neil McIntyre and Mirella Lapata. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572. Association for Computational Linguistics.

James Richard Meehan. 1976. The metanovel: writing stories by computer. Technical report, YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51.

Haoruo Peng and Dan Roth. 2016. Two discourse driven language models for semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 290–300, Berlin, Germany. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. PlotMachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295, Online. Association for Computational Linguistics.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Mark O Riedl. 2010. Story planning: Creativity through exploration, retrieval, and analogical transformation. *Minds and Machines*, 20(4):589–614.

Mark O Riedl and Robert Michael Young. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J Martin, Animesh Mehta, Brent Harrison, and Mark O Riedl. Controllable neural story plot generation via reward shaping.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J Martin, Animesh Mehta, Brent Harrison, and Mark O Riedl. 2019. Controllable neural story plot generation via reward shaping. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5982–5988. AAAI Press.

Bowen Tan, Zichao Yang, Maruan AI-Shedivat, Eric P Xing, and Zhiting Hu. 2020. Progressive generation of long text. *arXiv preprint arXiv:2006.15720*.

Peter Weyhrauch. 1997. *Guiding interactive fiction*. Ph.D. thesis, Ph. D. Dissertation, Carnegie Mellon University.

Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4306–4315.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.

BERT-based distractor generation for Swedish reading comprehension questions using a small-scale dataset

Dmytro Kalpakchi
Division of Speech, Music and Hearing
KTH Royal Institute of Technology
Stockholm, Sweden
dmytroka@kth.se

Johan Boye
Division of Speech, Music and Hearing
KTH Royal Institute of Technology
Stockholm, Sweden
jboye@kth.se

Abstract

An important part when constructing multiple-choice questions (MCQs) for reading comprehension assessment are the *distractors*, the incorrect but preferably plausible answer options. In this paper, we present a new BERT-based method for automatically generating distractors using only a small-scale dataset. We also release a new such dataset of Swedish MCQs (used for training the model), and propose a methodology for assessing the generated distractors. Evaluation shows that from a student's perspective, our method generated one or more plausible distractors for more than 50% of the MCQs in our test set. From a teacher's perspective, about 50% of the generated distractors were deemed appropriate. We also do a thorough analysis of the results.

1 Introduction

Multiple-choice questions (MCQs) are widely used for student assessments, from high-stakes graduation tests to lower-stakes reading comprehension tests. An MCQ consists of a question (stem), the correct answer (key) and a number of wrong, but plausible options (distractors). The problem of automatically generating stems with a key has received a great deal of attention, e.g., see the survey by Amidei et al. (2018). By comparison, automatically generating distractors is substantially less researched, although Welbl et al. (2017) report that manually finding reasonable distractors was the most time-consuming part in writing science MCQs. Indeed, reasonable distractors should be grammatically consistent and similar in length compared to the key and within themselves.

Given the challenges above, we attempt using machine learning (ML) to aid teachers in creating distractors for reading comprehension MCQs. The problem is not new, however most of the prior work has been done for English. In this paper we propose the first such solution for Swedish (although the proposed method is novel even for English, to the best of our knowledge). The key contributions of this work are: proposing a BERT-based method for generating distractors using only a small-scale dataset, releasing SweQUAD-MC[1], a dataset of Swedish MCQs, and proposing a methodology for conducting human evaluation aimed at assessing the plausibility of distractors.

2 Background

2.1 BERT for NLG

Devlin et al. (2019) introduced BERT as the first application of the Transformer architecture (Vaswani et al., 2017) to language modelling. BERT uses only Transformer's encoder stacks (with multi-head self-attention, MHSA), while the NLG community relies more on Transformer's decoder stacks (with masked MHSA) for text generation, e.g., GPT (Radford et al., 2018). However, Wang and Cho (2019) showed that BERT is a Markov random field, meaning that BERT learns a joint probability distribution over all sentences of a fixed length, and one could use Gibbs sampling to generate a new sentence. The authors compared samples generated autoregressively left-to-right by BERT and GPT, and found the perplexity of BERT samples to be higher than GPT's (BERT samples are of worse quality), but the n-gram overlap between the generated texts and texts from the dataset to be lower (BERT samples are more diverse).

Liao et al. (2020) show a way to improve BERT's generation capabilities via changing the masking scheme to a probabilistic one at training time. *Probabilistically masked language models* (PMLMs) assume that the masking ratio r for each sentence is drawn from a prior distribution $p(r)$. The au-

[1]The dataset and implementation of our models are available in this GitHub repository

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 387–403,
Aberdeen, Scotland, UK, 20-24 September 2021.

Property	Training	Development	Test
# of texts	434	64	45
# of MCQs	962	126	102
# of D	2.1 ± 0.5	2.1 ± 0.4	2.0 ± 0.2
Len(Text)	384.9 ± 330.1	355.1 ± 233.1	357.9 ± 254.3
Len(A)	4.2 ± 3.4	4.4 ± 3.5	4.6 ± 4.5
Len(D)	4.5 ± 3.9	4.3 ± 4.0	4.0 ± 3.7
|Len(A) - Len(D)|	1.9 ± 2.4	1.9 ± 2.3	1.9 ± 2.9

Table 1: Descriptive statistics of SweQUAD-MC dataset splits. A denotes the key, D denotes a distractor, Len(X) denotes a length of X in words. $x \pm y$ shows mean x and a standard deviation y

thors proposed to train a PMLM with a uniform prior (referred to as u-PMLM). The absence of the left-to-right restriction allows the model to generate sequences in an word arbitrary order. In fact, Liao et al. (2020) propose to generate sentences by randomly selecting the masked position, predicting a token for it, replacing the masked token with the predicted one and repeating the process until no masked tokens are left. The authors showed that the perplexity of the texts generated by u-PMLM is comparable to the ones by GPT.

2.2 Convolution partial tree kernels

As mentioned previously, plausible distractors should be grammatically consistent with the key. Hence, a metric measuring grammatical consistency would be useful both for quantitative evaluation and as a basis for a baseline method. We propose to use convolution partial tree kernels (CPTK) for these purposes. CPTK were proposed by Moschitti (2006) for dependency trees and essentially calculate the number of common tree structures (not only full subtrees) between two given trees. However, CPTKs can not handle labeled edges and were applied to dependency trees containing only lexicals. Another solution, proposed by Croce et al. (2011) and used in this article, is to include edge labels, i.e., grammatical relations (GR), as separate nodes. A resulting computational structure is Grammatical Relation Centered Tree (GRCT), which transforms the original dependency tree by making each PoS-tag a child of a GR node and a father of a lexical node. CPTKs can take any non-negative values and are thus hard to interpret. Hence, we use normalized CPTK (NCPTK) shown in Equation (1), where $K(T_1, T_2)$ is the CPTK applied to the dependency trees T_1 and T_2.

$$\widetilde{K}(T_1, T_2) = \frac{K(T_1, T_2)}{\sqrt{K(T_1, T_1)}\sqrt{K(T_2, T_2)}}, \quad (1)$$

Evidently, when T_1 and T_2 are the same, $\widetilde{K}(T_1, T_2)$ equals to 1, which is the highest value it can take.

3 Data

We have collected a Swedish dataset, henceforth referred to as *SweQUAD-MC*, consisting of texts and MCQs for the given texts. The dataset was created by three paid linguistics students instructed to pose unambiguous and independent questions. They were also asked to identify the key with at least two distractors, all of which are contiguous phrases in a given text. Additionally, as the distractors were required to be in the same grammatical form as the key (e.g., both in plural), the students were allowed to change the grammatical form of phrases if they constituted plausible distractors after this change. The exact instructions given to the students along with more details on the used texts are provided in Appendix A.

Each datapoint in SweQUAD-MC consists of a base text and an MCQ, i.e. a stem, the key and at least two distractors. The same text can be reused for different MCQs, but the sets of texts in training ($\sim 80\%$), development ($\sim 10\%$) and test ($\sim 10\%$) datasets are disjoint. However, some overlap in sentences is possible, since the texts might come from the same source. Descriptive statistics of all SweQUAD-MC splits is provided in Table 1.

4 Method

Given the small scale of SweQUAD-MC we have decided to fine-tune a pretrained BERT[2] for Swedish (Malmsten et al., 2020) on the task of distractor generation (DG). For achieving this, we have added on top of BERT two linear layers with layer normalization (Ba et al., 2016) in the middle to be trained from scratch (see architecture in Figure 1). The last linear layer is followed by a

[2]bert-base-cased

softmax activation giving probabilities over the tokens in the vocabulary for each position in the text. We trained the model using cross-entropy loss only for tokens in masked positions.

Recall that each MCQ consists of a base text `T`, the stem `Q` based on `T`, the key `A` and (on average) two distractors `D1` and `D2`. The DG problem is then to generate distractors conditioned on the context, consisting of `T`, `Q` and `A`. We provide all context components as input to the BERT model, separated from each other by the special separator token `[SEP]`. Given that BERT's maximum input length is 512 tokens, we trim `T` to the first 384 tokens (later referred to as `T_384`), since that is the average text length of the training set.

We have explored two different solution variants of DG. The first variant aims at generating distractors autoregressively, left to right. At generation time, the input to BERT consists of a context `CTX` (`T_384`, `Q` and `A` separated by `[SEP]` token), a `[SEP]` token, and a `[MASK]` token at the end. After a forward pass through BERT, the `[MASK]` token gets replaced by the word with the highest softmax score, which becomes the first word of the first distractor (dubbed `D11`). The generation of the first distractor continues by appending a `[MASK]` token after each forward pass until the network generates a separator token `[SEP]`, which concludes the generation of the first distractor `D1`. The next distractor `D2` is generated in the same way, except that the `CTX` is extended by `D1`. At training time, we use the same procedure, but with teacher forcing, allowing us to use the correct distractor tokens as targets for the cross-entropy loss (see example training datapoints for one MCQ in Table 2).

The second variant is inspired by u-PMLM, and aims at generating distractors autoregressively, but in an arbitrary word order. At generation time, the input to BERT consists of a context `CTX`, a `[SEP]` token, and a predefined number of `[MASK]` tokens (see Section 6.1). The generation proceeds by unmasking the token at the position where the model is most confident. This differs from unmasking a random position, proposed by Liao et al. (2020). The training procedure largely follows a masking scheme employed by u-PMLM by drawing the masking ratio from the uniform distribution (see example training datapoints for one MCQ in Table 2). Note that we do not include the `[SEP]` token when training, since we found that the trained model would constantly generate `[SEP]` tokens.

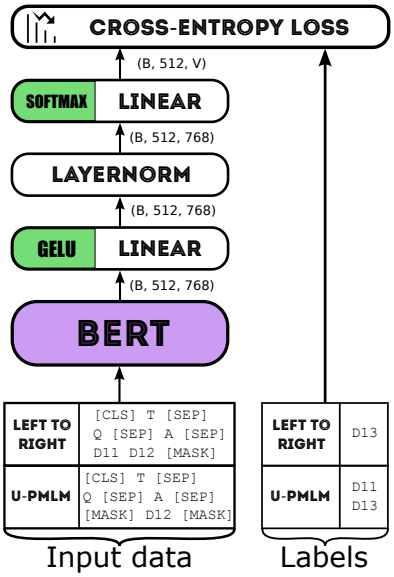


Figure 1: The DG model architecture. B is the batch size and V is the vocabulary size. The light green blocks represent the activation functions for the respective linear layers. The purple block represents parts of the network initialized with the pretrained weights.

Each sampled masking ratio r for the u-PMLM variant means that each token in the distractors from the dataset has a probability r to be masked. Hence, different r will potentially result in different number of masked tokens and at different positions. The number of times we draw r per distractor `DX` is proposed to be $\min(\text{Len}(\texttt{DX}), \texttt{MAX\_MASKINGS})$.

4.1 Baseline

As mentioned in Section 2.2, NCPTK measures grammatical consistency between the key and a distractor. Our baseline uses NCPTK on Universal Dependencies (UD) trees (Nivre et al., 2020) in the following way. For each given MCQ, we exclude the sentence containing the key from the base text and then parse each remaining sentence s_i of the text, and the key using the UD parser for Swedish. Let T_{s_i} and T_k denote a dependency tree corresponding to s_i and the key respectively. For each T_{s_i}, we find all subtrees with the root having the same universal PoS-tag and the same universal features (representing morphological properties of the token) as the root of T_k. If no subtrees are found, no distractors can be suggested for this MCQ. Otherwise, we calculate NCPTK between each found subtree and T_k (both as GRCT, but without lexicals). Then we take the textual representation of the K subtrees with the highest NCPTK as the distractor suggestions.

Input for left-to-right variant	Target
[CLS] CTX [SEP] [MASK]	D11
[CLS] CTX [SEP] D11 [MASK]	D12
[CLS] CTX [SEP] D11 D12 [MASK]	[SEP]
[CLS] CTX [SEP] D11 D12 [SEP] [MASK]	D21
[CLS] CTX [SEP] D11 D12 [SEP] D21 [MASK]	D22
[CLS] CTX [SEP] D11 D12 [SEP] D21 D22 [MASK]	D23
[CLS] CTX [SEP] D11 D12 [SEP] D21 D22 D23 [MASK]	[SEP]

Input for u-PMLM variant	Target(s)
[CLS] CTX [SEP] D11 [MASK]	D12
[CLS] CTX [SEP] [MASK] D12	D11
[CLS] CTX [SEP] D11 D12 [SEP] D21 [MASK] [MASK]	D22, D23
[CLS] CTX [SEP] D11 D12 [SEP] D21 [MASK] D23	D22
[CLS] CTX [SEP] D11 D12 [SEP] [MASK] D22 [MASK]	D21, D23

Table 2: Example datapoints extracted from one MCQ if training the autoregressive left-to-right variant (top table) or u-PMLM variant (bottom table). D1 and D2 are distractors, assumed to have 2 and 3 words, respectively. CTX represents the context, i.e., the sequence T_384 [SEP] Q [SEP] A, where T_384 is the first 384 tokens of the text, Q is a stem and A is the key.

5 Experimental setup

We have used Huggingface's Transformers library (Wolf et al., 2020) for implementing the DG model. The training hardware setup included 16 Intel Xeon CPU E5-2620 v4 (2.10GHz), 64 GB of RAM and 1 NVIDIA GeForce RTX 2080 Ti (11 GB VRAM). For this setup, we have fixed the random seed to 42, the number of training epochs to 6, the batch size to 4 (for both training and dev sets) and MAX_MASKINGS to 20 (for u-PMLM variant only). With these settings, training took about 3.67h for the left-to-right and 3h for the u-PMLM variant.

UD trees for the baseline were obtained using Stanza package (Qi et al., 2020) and convolution partial tree kernels on the UD trees were calculated using UDon2 library (Kalpakchi and Boye, 2020). Baseline requires no training and running our implementation of the baseline takes about a minute on the development or test set.

6 Evaluation

Following the analysis of Rodriguez (2005), we generate three distractors per MCQ for each model. Due to prohibitively high costs of human evaluation, we have divided the evaluation process into two stages. The first stage is quantitative evaluation, which gives limited information about the model's quality, but is sufficient for model selection. The second stage is human evaluation of the best model, selected during the first stage.

6.1 Quantitative evaluation

Automatic evaluation metrics, such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Denkowski and Lavie, 2014), CIDEr (Vedantam et al., 2015), became popular in NLG in recent years. Essentially, these metrics rely on comparing word overlap between a generated distractor and a reference one. Such metrics can yield a low score even if the generated distractor is valid but just happens to be different from the reference one, or a high score even though the distractor is ungrammatical but happens to have a high word overlap with the reference one (see the article by Callison-Burch et al. (2006) for a further discussion). Furthermore, they do not take into account how well a generated distractor is aligned with the key grammatically or how challenging the whole group of generated distractors would be.

To account for the properties mentioned above, we have experimented with a number of quantitative metrics and propose the following set to be used (the whole list is available in Appendix B). In the following list MCQ% means "Percentage of MCQ" and DIS means "generated distractor(s)".

1. *DisRecall*. Distractor recall.
2. *AnyDisRefMatch*. MCQ% with at least 1 DIS matching a reference one.
3. *AnyDisInText*. MCQ% with at least 1 DIS appearing in the base text.

4. *KeyInDis*. MCQ% with key being among DIS.

5. *AnySameDis*. MCQ% with ≥ 2 identical DIS.

6. *AllSameDis*. MCQ% with all identical DIS.

7. *AnyDisRep*. MCQ% with ≥ 1 DIS containing repetitive words contiguously.

8. *AnyDisEmpty*. MCQ% with ≥ 1 DIS being an empty string[3].

9. *AnyDisFromTrainDis*. MCQ% with at least 1 DIS matching with a distractor from training data, but not appearing in the base text.

10. *MeanNCPTK*, *MedianNCPTK*, *ModeNCPTK*. Mean, median, and mode NCPTK for pairs of UD trees for DIS and keys (all trees as GRCT, but ignoring nodes corresponding to lexicals).

The first group consists of metrics 1-3. The first two metrics count exact matches between generated and reference distractors. The rationale behind metric 3 is our assumption that distractors coming from the same text are more challenging. The higher the values of all these metrics are, the better.

The second group contains metrics 4-8, which give an idea of how challenging the whole group of distractors would be. For instance, duplicate distractors or ones with word repetitions could be excluded by students using common sense. The lower the metrics in this group are, the better.

The third group consists only of metric 9, serving as an overfitting indicator. The metric accounts for the distractors appearing as distractors in training data and high percentage indicates an overfitting possibility. The lower the values, the better.

The final group (item 10) measures how syntactically aligned generated distractors and the respective keys are. We employ NCPTK to measure the similarity of syntactic structures between each distractor and the respective key. Then we take mean, median and mode of the sequence of NCPTKs obtained in the previous step. The higher the values of these metrics are, the better.

Based on these metrics, we performed a model selection on the development set and chose the models performing best on the most of these metrics. Left-to-right model generated distractors token by token until either a `[SEP]` token was generated or the length of the distractor was 20 tokens.

[3]After excluding the special tokens, e.g., `[SEP]`

Metric	Baseline	u-PMLM
DisRecall ↑	1.44%	15.31%
AnyDisRefMatch ↑	2.94%	26.47%
AnyDisInText ↑	100.0%	72.55%
KeyInDis ↓	0.00%	4.9%
AnySameDis ↓	4.9%	13.73%
AllSameDis ↓	0.00%	1.96%
AnyDisRep ↓	0.00%	2.94%
AnyDisEmpty ↓	11.76%	0.00%
AnyDisFromTrainDis ↓	NA	0.98%
MeanNCPTK ↑	0.43	0.43
MedianNCPTK ↑	0.28	0.28
ModeNCPTK ↑	1.0 (20.56%)	1.0 (20.69%)

Table 3: Evaluation of DG models on the test set. When using u-PMLM, shortest distractors were generated first. ↑ (↓) means "the higher (lower), the better".

In contrast, u-PMLM needs the lengths of the distractors to be decided in beforehand, which we set to be the lengths of the two reference distractors and the length of the key[4]. Surprisingly, the order of distractors in terms of their length also matters for generation with u-PMLM, so we have tested three options: shortest first, longest first and random order. According to the results of model selection on the development set (presented in detail in Appendix C), u-PMLM models outperformed left-to-right models by a substantial margin.

The best u-PMLM model (generating shortest distractors first) and the baseline have been evaluated on the test set (see Table 3). Interestingly, the similarity of syntactic structures between the key and distractors (assessed by NCPTK) is the same for both baseline (that actually relies on NCPTK) and u-PMLM. At the same time, u-PMLM generates more distractors matching the reference ones compared to the baseline (as seen from *DisRecall* and *AnyDisRefMatch*). The baseline generates at least one empty string as a distractor 11.76% of the time (compared to no such cases for u-PMLM) limiting possibilities of using the baseline in the real-life applications.

6.2 Human evaluation

We have used distractors generated on the test set by the best u-PMLM model (selected after quantitative evaluation in Section 6.1) to conduct human

[4]If reference distractors are not available, we propose to generate distractors with the length differing by at most two words compared to the length of the key.

evaluation in 2 stages: from a perspective of a student and a teacher.

6.2.1 Student's perspective

A desirable property of reading comprehension MCQs is that the students should be unable to answer them correctly without reading the actual text. To put more formally, the average number of correctly answered MCQs without reading the actual text (denoted $\overline{N}_s$) should not differ significantly from the average number of correctly answered MCQs when choosing the answer uniformly at random (denoted $\overline{N}_r$). To test for this property, we have formulated the following two hypotheses.[5]

$$\mathcal{H}_0\text{: } \overline{N}_s = \overline{N}_r.$$
$$\mathcal{H}_1\text{: } \overline{N}_s \neq \overline{N}_r.$$

For N MCQs with 4 options, $\overline{N}_r = 0.25N$, which for our test set would be equal to $\overline{N}_r = 0.25 \cdot 102 = 25.5$. The appropriate statistical test in this case is one-sample two-tailed t-test with the aim of *not being able to reject* $\mathcal{H}_0$. Given that the purpose is to show that the data supports $\mathcal{H}_0$, we have set both the probability α of type I errors and the probability β of type II errors to be 0.05. Then we have used G*Power (Faul et al., 2009) to calculate the required sample size for finding a medium effect size (0.5) and the given α and β, which turned out to be 54 subjects.

Following the calculations above, we have recruited 54 subjects on the Prolific platform[6], and instructed them to choose the most plausible answer to a number of reading comprehension MCQs without providing the original texts. The collected data did not violate any assumptions for a one-sample t-test (see Appendix D.1 for more details). On average, the subjects correctly answered a significantly larger number of questions than $\overline{N}_r$ ($\overline{N}_s = 62.26$, $SE = 1.09$, $t(53) = 33.51, p < 0.05, r = 0.98$). To summarize, the chances of this sample to be collected are very low if $\mathcal{H}_0$ were true.

However, evidently some of the generated distractors were actually plausible, given that $\overline{N}_s \neq N$. To investigate the matter we have plotted the histogram of the frequency of choice of distractors by the subjects in Figure 2. As suggested by Haladyna and Downing (1993), distractors that are chosen by less than 5% of students should not be used, which in our case amounts to 39% of the distractors (the leftmost bar in Figure 2). If we eliminate these low-frequency distractors (LF-DIS), 68 MCQs (66.67%) will lose at least one distractor, 10 MCQs (9.8%) will lose all distractors and thus 34 MCQs (33.33%) will keep all 3 distractors.

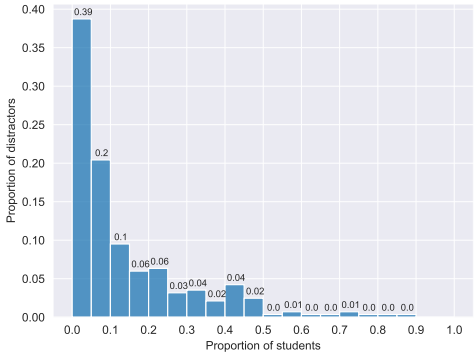


Figure 2: A histogram showing the frequency of choice of distractors in subjects' answers

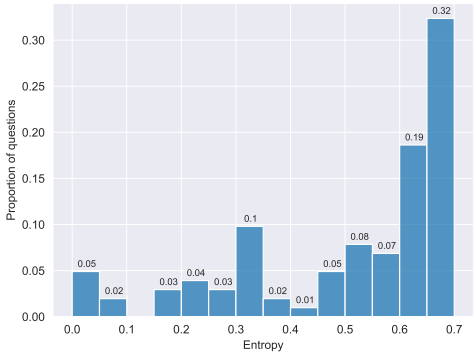


Figure 3: A histogram showing the entropy distribution per question

A more relaxed question is how many MCQs had at least one plausible distractor, which can be estimated by calculating the entropy for each question as shown in Equation (2), where A is the key, D is a distractor, Q is the stem, $P_Q(A)$ ($P_Q(D)$) is the probability that the key (any distractor) is chosen for Q by a subject.

$$H(Q) = -\sum_{O \in \{A,D\}} p_Q(O) \log(p_Q(O)) \quad (2)$$

The distribution of entropies per question is shown in Figure 3. Assuming the natural logarithm, the highest theoretically possible value for $H(Q)$ is 0.69, if $p_Q(A) = p_Q(D) = 0.5$. 32% of MCQs had an entropy larger than 0.65, whereas 51% had an entropy larger than 0.6, which means that half of MCQs had at least one plausible distractor.

[5] Preregistration is available here

[6] https://www.prolific.co/

6.2.2 Teacher's perspective

Bearing in mind the findings of Section 6.2.1, it is interesting to see which of the proposed distractors (especially, among LF-DIS) teachers would mark as acceptable. Given the complexity of such evaluation, using the whole test set was infeasible. To get a representative sample, we used entropy per question (shown in Figure 3). All MCQs were divided into 5 equally sized buckets by entropy and 9 MCQs were sampled uniformly at random from each bucket, resulting in 45 MCQs in total.

We asked 5 teachers to evaluate each MCQ (presented in a random order for each of them). Each MCQ contained the base text, the stem, the key and the generated distractors. The teachers were instructed to select those of generated distractors (if any) deemed suitable for testing reading comprehension. Additionally, we asked to provide their reasons for each rejected distractor in a free-text input. The inter-annotator agreement (IAA) was estimated using Goodman-Kruskal's γ (Goodman and Kruskal, 1979), specifically its multirater version γ_N proposed by Kalpakchi and Boye (2021). On the scale proposed by Rosenthal (1996), we have found a very large agreement ($\gamma_N = 0.85$, see Appendix D.2.2 for more details on IAA calculations).

On average, 1.47 distractors per MCQ were accepted by a teacher. Their reasons for rejections are distributed as shown in Figure 4. All teachers accepted at least one generated distractor for 39 MCQs (86.7%), whereas the majority of teachers did so for 27 MCQs (60%). Interestingly, there are no MCQs in which all 5 teachers have either accepted or rejected all generated distractors. However, the majority of teachers has accepted or rejected all distractors for 4 MCQs (8.9%) and 6 MCQs (13.3%) respectively.

Out of 45 MCQs, 31 (68.9%) had at least one LF-DIS, as defined in Section 6.2.1. For these 31 MCQs we report a distribution of accepted/rejected LF-DIS by the majority of teachers in Figure 5. Let us call the 15 MCQs with all LF-DIS accepted by the majority of teachers as *mismatch MCQs* (lowest row in Figure 5). Interestingly, 12 of the 15 mismatch MCQs had at least one more distractor in addition to LF-DIS being accepted by the majority of teachers. Furthermore, all mismatch MCQs had entropy higher than 0.3. This entails that almost a half of LF-DIS should *not* necessarily be thrown away, since they were accepted by teachers, but the MCQs either happened to have more plausible distractors or subjects might have had relevant background knowledge to answer the questions.

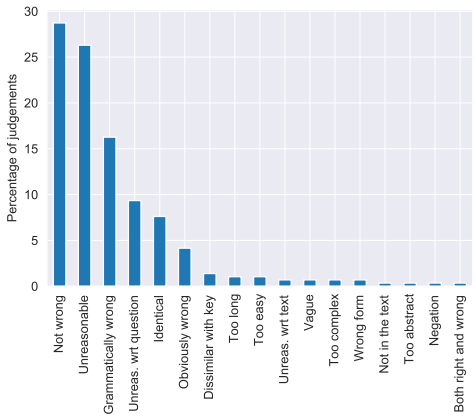


Figure 4: A histogram showing the distribution of teachers' reasons behind rejecting distractors.

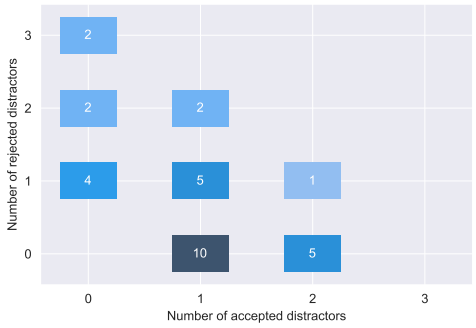


Figure 5: A bi-variate histogram showing the distribution of the 31 MCQs (the numbers on the bars sum to 31) with at least 1 LF-DIS, with respect to their LF-DIS being accepted/rejected by the majority of teachers.

7 Related work

We employed a systematic process to get a comprehensive overview of DG methods (see Appendix E for more details). Out of the resulting 28 articles (see an overview in Table 4), only 2 worked with a language other than English (Chinese and Basque). In this paper we work on reading comprehension MCQs, which makes only 12 papers, dealing with factual questions, relevant.

Two of these used rule-based approaches. Majumder and Saha (2015) generated MCQs for cricket domain and used a number of hand-crafted rules based on gazeteers and Wikipedia entries to generate distractors. Mitkov and Ha (2003) proposed to generate distractors for MCQs on electronic instructional documents using WordNet.

Six of these relied on extractive approaches.

Liang et al. (2018), Welbl et al. (2017), and Ha and Yaneva (2018) formulated choosing a distractor as a ranking problem from the given candidate set. In the first two articles the candidate set constituted all distractors from the available MCQ dataset. The authors then trained ML-based ranker(s) for choosing the best distractors. In the last one, the candidate set was created using content engineers. Distractors with a high similarity of their concept embeddings (summed for multiple words) and appearing in the same document as the key are ranked higher. Stasaski and Hearst (2017) and Araki et al. (2016) worked in the domain of biology. The former used an ontology and the latter employed event graphs containing information about coreferences to generate distractors. Karamanis et al. (2006) used thesaurus and tf-idf to identify key concepts in the given text and then select as distractors those having the same semantic type as the key.

The remaining four employed neural methods and are most relevant among the surveyed. Qiu et al. (2020) trained a sequence-to-sequence (seq2seq) model with a number of attention layers. Zhou et al. (2020) also employed a seq2seq model, but with a hierarchical attention to capture the interaction between a text and a question, as well as semantic similarity loss. Both articles used a beam search combined with filtering based on Jaccard coefficient at generation time. Offerijns et al. (2020) trained a GPT-2 model to generate 3 distractors for a given MCQ, and used BERT-based question answering model for quantitative evaluation (along with human evaluation).

Finally, Chung et al. (2020) proposed a BERT-based method for English with answer-negative regularization, penalizing distractors for containing the same words as the key, and training a sequential and a parallel MLM model simultaneously. At generation time, they generate one distractor, and then create a distractor set of the predefined size based on sampling from the probability distribution returned by BERT for each token of the distractor. Then they rank every triple of distractors based on the entropy of a separately trained QA model.

Problem/method property	**#**
■ Extractive	14
■ Generative, rule-based	7
■ Generative, neural	7
● Only automatic evaluation	5
● Only human evaluation	19
● Automatic and human evaluattion	4
▲ Cloze-style, single-word answers	14
▲ Cloze-style, continue the sentence	2
▲ Factual questions	12

Table 4: 28 related works broken down by method (■), type of evaluation (●) and types of questions for which distractors have been generated (▲)

Our method also relies on BERT, but has a number of differences beyond being applied to Swedish. Firstly, we did not include answer-negative regularization, since it is not always a good strategy. For instance, given the stem "When should you pay a fee if you apply for a visa?" and a key "before you have submitted the application", the best distractor would be "after you have submitted the application", which shares most of the words with the key. Secondly, we generate distractors in arbitrary word order compared to left-to-right generation in (Chung et al., 2020). Thirdly, at generation time, we use previously generated distractors as input for generating next ones, and always take tokens with a maximum probability. This lowers the risk of generating ungrammatical distractors. Finally, our training set is 100 times smaller compared to the training set used by Chung et al. (2020).

8 Conclusion

We have collected SweQUAD-MC, the first dataset of Swedish MCQs, and showed the possibility of training usable BERT-based DG models, despite the small scale of the dataset. We have showed that a u-PMLM variant of the BERT-based DG model performs best on the dataset, and proposed a novel methodology of evaluating the plausibility of generated distractors. Around half of the generated distractors were found acceptable by the majority of teachers, and more than 50% of MCQs had at least one plausible generated distractor, judging by the entropy of students' responses.

Bearing in mind that the aim of the proposed method is to support (not replace) teachers, we deem that our method works well for MCQs in Swedish (and potentially in other languages with a pretrained BERT and a dataset of a similar scale).

Furthermore, we have presented a baseline applicable to any language with a UD treebank (currently about 100 languages). Although its performance is nowhere near the u-PMLM variant, we believe that it can serve as a good point of comparison to emerging neural methods for other languages.

Acknowledgments

This work was supported by Vinnova (Sweden's Innovation Agency) within project 2019-02997. We would like to thank the anonymous reviewers for their comments, as well as Gabriel Skantze and Bram Willemsen for their helpful feedback prior to the submission of the paper.

References

Jacopo Amidei, Paul Piwek, and Alistair Willis. 2018. Evaluation methodologies in automatic question generation 2013-2018. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 307–317, Tilburg University, The Netherlands. Association for Computational Linguistics.

Jun Araki, Dheeraj Rajagopal, Sreecharan Sankaranarayanan, Susan Holm, Yukari Yamakawa, and Teruko Mitamura. 2016. Generating questions and multiple-choice answers using semantic analysis of texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1125–1136, Osaka, Japan. The COLING 2016 Organizing Committee.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of Bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Ho-Lam Chung, Ying-Hong Chan, and Yao-Chung Fan. 2020. A BERT-based distractor generation scheme with multi-tasking and negative answer training strategies. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4390–4400, Online. Association for Computational Linguistics.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

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

Franz Faul, Edgar Erdfelder, Axel Buchner, and Albert-Georg Lang. 2009. Statistical power analyses using g* power 3.1: Tests for correlation and regression analyses. *Behavior research methods*, 41(4):1149–1160.

Leo A Goodman and William H Kruskal. 1979. Measures of association for cross classifications. *Measures of association for cross classifications*, pages 2–34.

Le An Ha and Victoria Yaneva. 2018. Automatic distractor suggestion for multiple-choice tests using concept embeddings and information retrieval. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 389–398, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas M Haladyna and Steven M Downing. 1993. How many options is enough for a multiple-choice test item? *Educational and psychological measurement*, 53(4):999–1010.

Dmytro Kalpakchi and Johan Boye. 2020. UDon2: a library for manipulating Universal Dependencies trees. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 120–125, Barcelona, Spain (Online). Association for Computational Linguistics.

Dmytro Kalpakchi and Johan Boye. 2021. Quinductor: a multilingual data-driven method for generating reading-comprehension questions using universal dependencies. *arXiv preprint arXiv:2103.10121*.

Nikiforos Karamanis, Le An Ha, and Ruslan Mitkov. 2006. Generating multiple-choice test items from medical text: A pilot study. In *Proceedings of the Fourth International Natural Language Generation Conference*, pages 111–113, Sydney, Australia. Association for Computational Linguistics.

Chen Liang, Xiao Yang, Neisarg Dave, Drew Wham, Bart Pursel, and C. Lee Giles. 2018. Distractor generation for multiple choice questions using learning to rank. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 284–290, New Orleans, Louisiana. Association for Computational Linguistics.

Yi Liao, Xin Jiang, and Qun Liu. 2020. Probabilistically masked language model capable of autoregressive generation in arbitrary word order. In *Proceedings of the 58th Annual Meeting of the Association*

for Computational Linguistics, pages 263–274, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Mukta Majumder and Sujan Kumar Saha. 2015. A system for generating multiple choice questions: With a novel approach for sentence selection. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*, pages 64–72, Beijing, China. Association for Computational Linguistics.

Martin Malmsten, Love Börjeson, and Chris Haffenden. 2020. Playing with words at the national library of sweden–making a swedish bert. *arXiv preprint arXiv:2007.01658*.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pages 17–22.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning*, pages 318–329. Springer.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Jeroen Offerijns, Suzan Verberne, and Tessa Verhoef. 2020. Better distractions: Transformer-based distractor generation and multiple choice question filtering. *arXiv preprint arXiv:2010.09598*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Zhaopeng Qiu, Xian Wu, and Wei Fan. 2020. Automatic distractor generation for multiple choice questions in standard tests. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2096–2106, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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

Michael C Rodriguez. 2005. Three options are optimal for multiple-choice items: A meta-analysis of 80 years of research. *Educational measurement: issues and practice*, 24(2):3–13.

James A Rosenthal. 1996. Qualitative descriptors of strength of association and effect size. *Journal of social service Research*, 21(4):37–59.

Katherine Stasaski and Marti A. Hearst. 2017. Multiple choice question generation utilizing an ontology. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 303–312, Copenhagen, Denmark. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Alex Wang and Kyunghyun Cho. 2019. BERT has a mouth, and it must speak: BERT as a Markov random field language model. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 30–36, Minneapolis, Minnesota. Association for Computational Linguistics.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing:*

System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Xiaorui Zhou, Senlin Luo, and Yunfang Wu. 2020. Co-attention hierarchical network: Generating coherent long distractors for reading comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9725–9732.

A SweQUAD-MC data collection details

We have used publicly available texts from the websites of Swedish government agencies. The exact list of URLs is provided in the GitHub repository associated with the paper. The exact instructions given to students recruited to collect SweQUAD-MC dataset (and their translation to English) are presented in Figure 6. In addition to the given instructions, the students were also given the opportunity to slightly reformulate the distractors found in the text in order to align the syntactic structure with that of the key.

B Quantitative metrics

In addition to the metrics 1–10 presented in Section 6.1, we have also looked at the following ones (MCQ% means "Percentage of MCQ" and DIS means "generated distractor(s)")

11. MCQ% with at least 1 DIS being capitalized differently from the key
12. MCQ% with at least 1 DIS being a distractor from training data.
13. MCQ% with at least 1 DIS is in any base text from training data.
14. MCQ% with at least 1 DIS appearing in at least 1 base text from training data, but not in their own base text.
15. MCQ% with all distractors appearing in the base text.
16. MCQ% with all distractors appearing in at least 1 base text from training data.
17. MCQ% with all DIS being distractors from training data.

The rationale behind metric 11 was that capitalized answers are named entities and thus one would like distractors also to be named entities. However, it does not always hold. For instance, consider the stem "Who gets an e-mail with a confirmation of a successful submission of the application for the work permit?" and the key "you and your employer". A distractor "Migration Agency" would suit the question perfectly, although capitalization is clearly different.

Metrics 12-17 were candidates to become overfitting indicators. However, metric 2 was excluded, since *AnyDisFromTrainDis* is more informative, given phrases used as distractors in training data can be repeated in other texts. Metrics 13-14 were excluded, since it's unclear whether the higher or lower values are better. For instance, if a text from the training data and the given text are thematically similar, would copying a distractor from training data be considered overfitting? Metrics 15-17 were rejected as too strict, leaving the possibility of actually missing overfitting if only 2 of 3 distractors would meet the criteria.

C Model selection

We have trained both left-to-right and u-PMLM variants for 6 epochs (fixing a random seed for u-PMLM masking procedure to 42). The quantitative performance metrics on the development set for the top-3 models for each variant are presented in Table 5. The best u-PMLM model (i-14000) outperformed the best left-to-right model (i-18000) on most of the quantitative metrics.

The next experiment concerned the order in which distractors are generated, which we tested only for the best u-PMLM model. We tried generating shortest distractors first (SF), longest first (LF) or in a random order with a fixed seed of 42 (RND). The results of the experiment are presented in Table 6. Evidently, models with SF-generation consistently outperform ones with LF-generation. SF-generation also performs on-par or better than RND-generation. However, fixing a seed is not a generalizable solution, which is why we opted for SF-generation.

D Human evaluation details

D.1 Student's perspective

Evaluation from the student's perspective has been conducted on the Prolific platform[7]. We used Prolific's pre-screening feature and required each subject to have Swedish as the first language and hold at least a high school diploma (A-levels). Descriptive statistics about the recruited sample of subjects

[7] https://www.prolific.co/

Imagine that you are a teacher checking reading comprehension skills of your students. Given a text, your task is to create one or more multiple choice questions based on the text, i.e.:

1. formulate a question with the correct answer in the text;
2. mark the correct answer in the text;
3. mark some wrong, but plausible options in the text.

When you have written your questions, marked the correct answer (CA) and the wrong alternatives in the text, click on "Submit". When you formulate the question, think about the following aspects.

- The question must be independent, i.e., one should not require additional information (on top of the given text) to be able to answer the question.
- The question should be unambiguous and have only one possible interpretation.
- One should not be able to answer your question without reading the text, which is why even wrong alternatives should be plausible.
- Wrong options must be in the same grammatical form as the CA. For instance, if the CA begins with a verb in Past Simple, all wrong options must begin with a verb in Past Simple.

Find as many questions as you can (+ the correct answer and wrong alternatives) on each text and then get a new text when you can't find more.

Figure 6: An English translation of the original instructions for SweQUAD-MC data collection (the original instructions in Swedish can be found in the GitHub repository)

Metric	**left-to-right**			**u-PMLM**		
	i-10000 e-3.02	**i-14000 e-4.23**	**i-18000 e-5.43**	**i-10000 e-3.59**	**i-14000 e-5.02**	**i-16000 e-5.74**
M1: DisRecall ↑	9.77%	14.29%	12.41%	17.67%	**21.43%**	18.80%
M2: AnyDisRefMatch ↑	18.25%	26.19%	21.43%	30.95%	**37.30%**	31.75%
M3: AnyDisInText ↑	64.29%	69.84%	**73.81%**	68.25%	72.22%	73.81%
M4: KeyInDis ↓	0.79%	1.59%	3.17%	2.38%	5.56%	5.56%
M5: AnySameDis ↓	34.13%	27.78%	**19.84%**	9.52%	10.32%	11.90%
M6: AllSameDis ↓	3.17%	1.59%	**0.79%**	1.59%	**0.79%**	0.79%
M7: AnyDisRep ↓	0.00%	0.00%	0.00%	0.00%	1.59%	1.59%
M8: AnyDisEmpty ↓	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
M9: AnyDisFromTrainDis ↓	5.56%	5.56%	6.35%	5.56%	**2.38%**	2.38%
M10: MeanNCPTK ↑	0.33	0.38	**0.39**	0.41	0.41	0.41
M11: MedianNCPTK ↑	0.18	0.19	**0.21**	0.27	0.26	0.27
M12: ModeNCPTK ↑	1.0 (13.3%)	1.0 (18.8%)	1.0 (17.6%)	1.0 (18.1%)	**1.0 (20.3%)**	1.0 (19.6%)

Table 5: TOP-3 models for left-to-right and u-PMLM variants after model selection on the dev set. i-XXXXX shows a number of iterations since training start, e-X.XX shows a number of epochs corresponding to i-XXXXX. Floating point epochs are due to checkpoints being saved every 2000 iterations.

Metric	i-10000, e-3.59			i-14000, e-5.02			i-16000, e-5.74		
	SF	**LF**	**RND**	**SF**	**LF**	**RND**	**SF**	**LF**	**RND**
M1 ↑	15.8%	13.9%	15.8%	20.7%	14.7%	19.9%	19.9%	15.0%	17.7%
M2 ↑	25.4%	25.4%	29.4%	36.5%	27.8%	34.1%	34.1%	27.0%	30.1%
M3 ↑	64.3%	63.5%	65.9%	73.0%	66.7%	69.8%	72.2%	66.7%	70.6%
M4 ↓	2.4%	2.4%	3.2%	4.0%	4.8%	5.6%	4.8%	5.6%	4.8%
M5 ↓	7.9%	11.1%	7.9%	10.3%	9.5%	10.3%	10.3%	8.7%	10.3%
M6 ↓	1.6%	1.6%	1.6%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%
M7 ↓	0.0%	1.6%	0.0%	0.0%	1.6%	1.6%	0.8%	0.8%	3.2%
M8 ↓	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
M9 ↓	5.6%	4.8%	6.3%	4.8%	5.6%	4.0%	4.0%	4.0%	3.2%
M10 ↑	0.41	0.41	0.41	0.41	0.41	0.41	0.41	0.41	0.41
M11 ↑	0.24	0.22	0.25	0.26	0.21	0.22	0.29	0.22	0.22
M12 ↑	1.0 (18%)	1.0 (17%)	1.0 (19%)	1.0 (20%)	1.0 (18%)	1.0 (20%)	1.0 (19%)	1.0 (18%)	1.0 (19%)

Table 6: Results of model selection by the generation order of distractors for the TOP-3 u-PMLM models.

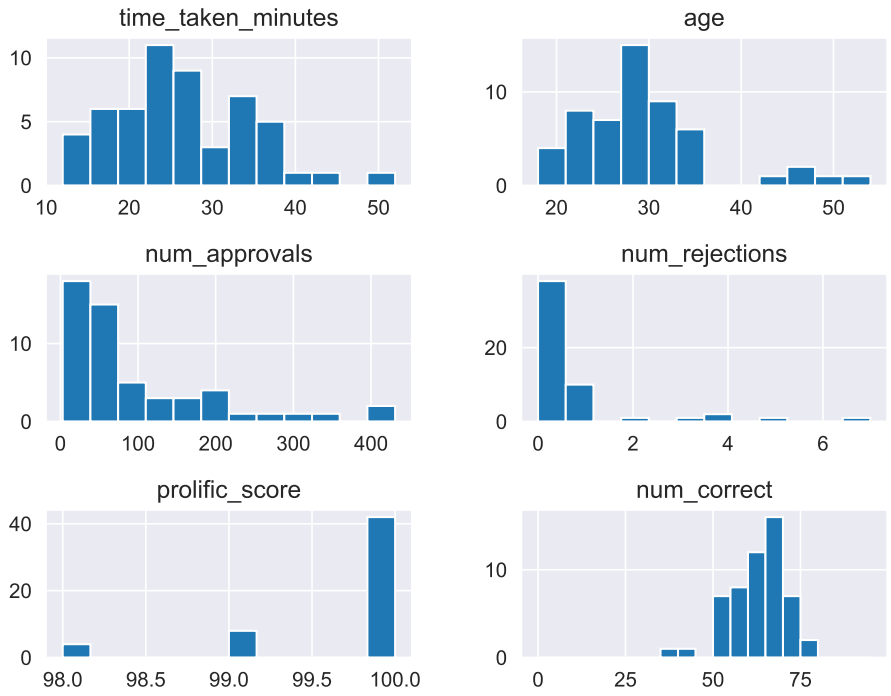


Figure 7: Descriptive statistics of the sample of subjects on Prolific

Thank you for participating in our study! You will be presented with a number of multiple choice questions. Your task is to answer as many of these questions correctly as possible. If you don't know which alternative is correct, choose the one that seems the most plausible. You are allowed to use **ONLY** your own prior knowledge and common sense. Please, do **NOT** consult any other external sources of information.

Figure 8: An English translation of the original instructions given to subjects on the Prolific platform (the original instructions in Swedish can be found in the GitHub repository)

is presented in Figure 7. The exact guidelines given to the subjects (and their translation to English) are presented in Figure 8. MCQs were presented in a random order, but the order of options for each MCQs was the same for each subject.

D.1.1 Check of the t-test assumptions

We used one sample t-test for conducting our analysis and thus the following assumptions were checked for.

1. **The variable under study should be either an interval or ratio variable**. Our variable, the number of correctly answered MCQs, is clearly on a ratio scale.

2. **The observations in the sample should be independent**. Subjects have performed the task independently of each other through a Prolific platform, hence the observations are independent.

3. **The variable under study should be approximately normally distributed**. The distribution of the number of correctly answered MCQs is presented in Figure 7 (the plot in the last row and the last column with the title "num_correct"). The distribution is indeed approximately normal.

4. **The variable under study should have no extreme outliers**. Outliers are typically defined in terms of the interquartile range (IQR), which equals to Q3 - Q1. The datapoints outside 1.5IQR are deemed mild outliers, whereas those outside 3IQR are considered extreme outliers. Boxplots for our data with whiskers within both 1.5IQR and 3IQR are presented in Figure 9. Two datapoints can be considered mild outliers, but no extreme outliers are present, which means this assumption for the one sample t-test is not violated.

D.2 Teacher's perspective

D.2.1 Instructions

The exact guidelines given to the teachers and their translation to English, are presented in Figure 10.

D.2.2 Inter-annotator agreement

To evaluate the inter-annotator agreement (IAA) between the teachers, we have reformulated the problem into a ranking problem, where all accepted distractors were given the rank of 1 and those rejected - the rank of 2. IAA was then estimated using Goodman-Kruskal's γ (Goodman and Kruskal, 1979), specifically its multirater version γ_N proposed by Kalpakchi and Boye (2021). The total number of concordant and discordant pairs were summed for each pair of teachers for each MCQ. The resulting γ_N equals to 0.85, indicating a very large agreement on the scale proposed by Rosenthal (1996).

E Details on surveying related work

To get a comprehensive overview of methods for generating distractors for MCQs, we employed a two-step process. The first step was to issue queries "distractor generation" and "multiple choice question generation" to ACL Anthology and Google Scholar. The result was 20 articles from ACL Anthology and 4 additional ones from Google Scholar. The second step was to select relevant references from the "Related work" sections of these articles. This resulted into 15 additional articles. Out of found 39 articles, 11 were filtered out (8 focused only on generating questions, 1 relied mostly on expert knowledge, 1 on the auxiliary relation extraction task and 1 was a demo paper), leaving 28 articles in total. Only 2 of these 28 papers worked with a language other than English (Chinese and Basque).

F Generated samples

A number of generated distractors along with the respective stems and keys from the dataset are presented in Figures 11, 12, 13, 14, 15. The questions are sampled based on the entropy of student's an-

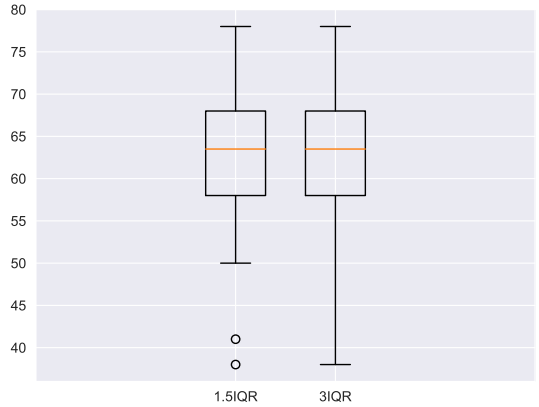


Figure 9: Boxplots for the number of correctly answered questions

Thank you for participating in our study! You will be presented with a number of tests. Each test contains a text, a reading comprehension question based on the text, the explicitly marked correct answer to this question and a number of suggestions for wrong, but plausible alternatives (distractors).
Suppose you would like to use the given question for testing reading comprehension of the given text. Your task is to judge which of the suggested distractors (if any) you would fit the purpose. Select suitable distractors by simply ticking the respective checkboxes. For the other distractors (that you didn't select), please briefly state your reasons why these distractors were inappropriate in the respective text fields (max 1 sentence).

Figure 10: An English translation of the original instructions given to teachers (the original instructions in Swedish can be found in the GitHub repository)

swers using the same 5 buckets as in sampling for teachers' evaluation. Recall that distractors are said to be low frequency (LF-DIS) if they were chosen by less than 5% of students. Hence, a red cross in the column "F-DIS $> 5\%$" entails that a given distractor is in fact an LF-DIS.

The MCQ in sample 1 has an entropy of 0, meaning all students have selected the same option, which in this case was the key. In this case, two of three distractors were accepted by the majority of teachers, although all of them were LF-DIS. This is a good example of an MCQ with plausible distractors, but where the stem is too easy.

The MCQ in sample 2 presents an interesting case, when the distractor contains an obvious grammatical error (comma before the first word in the distractor 3). While the distractor was rightfully rejected by the majority of teachers, it was still selected by more than 5% of students.

The MCQ in sample 3 is a good example of longer distractors. In this case, two distractors were accepted by teachers and two were selected by more than 5% of students. However, interestingly these sets are disjoint, meaning that all three distractors could potentially be useful. Another more general observation, requiring future research, is that our model seems to struggle more when generating longer distractors in general, resulting in non-finished sentences or repetitions of words.

The MCQ in sample 4 is somewhat opposite to sample 3, since one distractor that was accepted by the teachers turned out to be an LF-DIS. This either means that the stem was too easy or that none of the distractors were potentially useful.

The MCQ in sample 5 is the one with a highest theoretically possible entropy between selecting the correct or a wrong option. Note that it might still happen that some of the distractors is LF-DIS, since the entropy is calculated not between all four options, but only between the key and the distractors as a group.

Stem
Vad täcker över hälften av Sveriges yta?
(*What covers more than half of the surface of Sweden?*)

Key: skog (*forest*)

Distractor (sv)	**Distractor (en)**	**Accepted by teachers?**	**F-DIS** $> 5\%$
vattendrag	*water*	✔	✘
miljöer	*environments*	✘	✘
djur - och växtarter	*plant and animal species*	✔	✘

Figure 11: Sample 1 (entropy 0). "F-DIS" denotes the frequency of choice of a distractors by the students, "Accepted by teachers" indicates if a distractor was accepted by the majority of teachers.

Stem
Vad förvaras på en torkanläggning?
(*What is stored in a drying facility?*)

Key: spannmål, hö eller halm (*grains, hay or straw*)

Distractor (sv)	**Distractor (en)**	**Accepted by teachers?**	**F-DIS** $> 5\%$
ogräs	*weeds*	✔	✘
balpressar	*balers*	✔	✔
, harvar och sår	*, harrows and sows*	✘	✘

Figure 12: Sample 2 (entropy 0.31). "F-DIS" denotes the frequency of choice of a distractors by the students, "Accepted by teachers" indicates if a distractor was accepted by the majority of teachers.

Stem
När betalar du avgiften om du ansöker på en ambassad?
(When do you pay the fee when you are applying at an embassy?)

Key: när du lämnar in din ansökan (*when you are handing in your application*)

Distractor (sv)	**Distractor (en)**	**Accepted by teachers?**	**F-DIS** $> 5\%$
när du får uppehållstillstånd	*when you receive your residence permit*	✔	✔
när du ansöker för första gången	*when you are applying for the first time*	✘	✔
när du ansöker innan tiden för ditt tidigare tillstånd har gått ut	*when you are applying before your previous permit has expired*	✔	✘

Figure 13: Sample 3 (entropy 0.57). "F-DIS" denotes the frequency of choice of a distractors by the students, "Accepted by teachers" indicates if a distractor was accepted by the majority of teachers.

Stem
Vad är negativt för flera marina miljöer?
(*What is negative for several marine environments?*)

Key: kommersiellt fiske (*commercial fishing*)

Distractor (sv)	**Distractor (en)**	**Accepted by teachers?**	**F-DIS** $> 5\%$
klimatförändringar	*climate change*	✘	✔
många olika miljöer	*many different environments*	✘	✘
äldre brukningsmetoder	*older cultivation methods*	✔	✘

Figure 14: Sample 4 (entropy 0.675). "F-DIS" denotes the frequency of choice of a distractors by the students, "Accepted by teachers" indicates if a distractor was accepted by the majority of teachers.

Stem
Vilka kan utfärda medicinska rapporter för kabinbesättning?
(*Who can issue medical reports for cabin crew?*)

Key: företagsläkare (*company physicians*)

Distractor (sv)	**Distractor (en)**	**Accepted by teachers?**	**F-DIS** $> 5\%$
företagssköterskor	*company nurses*	✔	✔
flygläkare	*aviation physicians*	✘	✔
gymnasieinfo.se	*gymnasieinfo.se*	✘	✘

Figure 15: Sample 5 (entropy 0.69). "F-DIS" denotes the frequency of choice of a distractors by the students, "Accepted by teachers" indicates if a distractor was accepted by the majority of teachers.

Exploring Structural Encoding for Data-to-Text Generation

Joy Mahapatra
Indian Statistical Institute, Kolkata
joymahapatra90@gmail.com

Utpal Garain
Indian Statistical Institute, Kolkata
utpal@isical.ac.in

Abstract

Due to efficient end-to-end training and fluency in generated texts, several encoder-decoder framework-based models are recently proposed for data-to-text generations. Appropriate encoding of input data is a crucial part of such encoder-decoder models. However, only a few research works have concentrated on proper encoding methods. This paper presents a novel encoder-decoder based data-to-text generation model where the proposed encoder carefully encodes input data according to underlying structure of the data. The effectiveness of the proposed encoder is evaluated both extrinsically and intrinsically by shuffling input data without changing meaning of that data. For selecting appropriate content information in encoded data from encoder, the proposed model incorporates *attention gates* in the decoder. With extensive experiments on WikiBio and E2E dataset, we show that our model outperforms the state-of-the models and several standard baseline systems. Analysis of the model through component ablation tests and human evaluation endorse the proposed model as a well-grounded system.

1 Introduction

Data-to-text generation (Gatt and Krahmer, 2018; Reiter and Dale, 2000) aims to produce human-understandable text from semi-structured data such as tables, concepts, etc. The input data consists of multiple records (or events), where each record represents a particular field (or attribute) of the data. Table 1 shows an example for data-to-text generations, where text y is used to describe restaurant data x . The example consists of three records, and each record describes a field with a name (underlined part) and corresponding values (*italic* part).

data (x)

record 1	name: *The Punter*
record 2	food: *English*
record 3	price range: *high*

text (y)

The Punter is a restaurant with high prices.

Table 1: An example of data-to-text generation.

Due to efficient end-to-end training and fluency in generated texts (Novikova et al., 2017b; Sutskever et al., 2014), numerous data-to-text generation systems have adopted encoder-decoder based model (Gong et al., 2019; Liu et al., 2018). In such encoder-decoder based models, a proper meaningful encoding of input data is a real concern. As in most of the data-to-text generation, input data poses record-field structure (like in table 1), it is natural to realise the need for appropriate *structural* encoding for record-field structure. Most of the existing encoder-decoder models for data-to-text generation (Liu et al., 2018; Nema et al., 2018) primarily focus more on attention mechanisms than structural encoding. However, a recent interesting study by Gong et al. (2019) shows some effective encoding strategies based on functional dimensions of data.

Selecting appropriate content from input data (also known as content selection) is an important task. For example, according to table 1, the 'name' and 'price range' fields of the data is considered as contents w.r.t. text y. Detecting such contents from encoded data is a hard task.

We propose an encoder-decoder model for data-to-text generation where the encoder encodes data based on record-field structures. We introduce structure-wise attention gates to capture appropriate content from the encoded data. As records in an input data don't pose any ordering among them

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 404–415,
Aberdeen, Scotland, UK, 20-24 September 2021.

(for example, in table 1, there is no order among the three records in x), the efficiency of the proposed encoder is estimated through shuffling those records. The comparisons of our system with existing high-scoring systems on WikiBio and E2E dataset bring out the distinction of our model. Additional human evaluation signifies that the proposed model performs well in terms of both readability and adequacy of generated text.

2 Notations

We follow the notations which are commonly used in popular data-to-text generation models (Angeli et al., 2010; Nie et al., 2019; Dhingra et al., 2019). The goal of our task ($\mathcal{T}$) is to produce text representation $y = y_1 y_2 ... y_w$ (where y_i is the i-th word in y) from a given an input data x.

$$\mathcal{T} \; : \; \hat{y} \leftarrow \underset{y}{\text{argmax}} \; p(y|x)$$

An input (x) consists of multiple records $\{r_i\}_{i=1}^{n}$. A record (r_j) contains a field f^j, with its name n^j and the corresponding value $v^j = v_1^j v_2^j ... v_w^j$ (where, v_i^j is the i-th word in v^j).

3 Approach

We propose a neural network based encoder-decoder model for the task $\mathcal{T}$, where the encoder structurally encodes the input (x). The decoder is a recurrent neural network with two sub-modules—(i) attention gates for appropriate content selection and (ii) copy module to handle the appearances of rare words in generating text.

3.1 Structural Encoder

The notion of the proposed encoder comes from the underlying structure of input data. We consider each input data comprises of two structures—(i) fields as fine-grained structure; (ii) records as coarse-grained structure. For example, in figure 1, input data x contains two records ($\{r_i\}_{i=1}^{2}$) with each record consists of two field parts ($r_1 = (f_1^1, f_2^1)$ and $r_2 = (f_1^2, f_2^2)$).

The proposed encoder encodes input data based on this record-field structures (Figure 1) in bottom-up approach. Each structure (field and record) encoding involves two types of connections (the arrows in figure 1 show these connections)—

- The horizontal dotted arrows denote **horizontal connections**—the objective of these connections is to help in making relationships among close components.
- The dashed arrows denote **hierarchical connections**—the purpose of these connections is to accumulate information from all similar components (either records or fields) and forward that information to next stage.

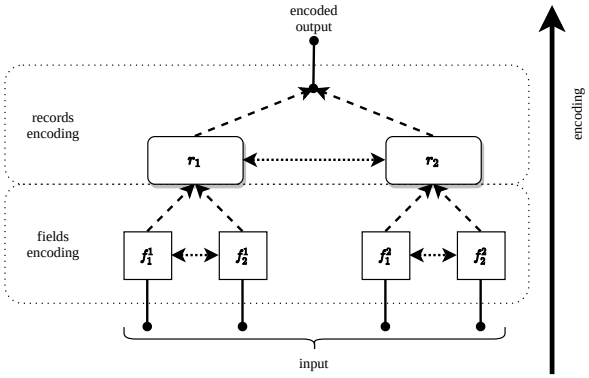


Figure 1: Structure of input data and bottom-up encoding.

So with the proposed structural encoder, knowledge about the record and field structures of input data get encoded.

3.1.1 Field Encoding

In field encoding of our structural encoder, all field words inside of a record are encoded together into a single vector representation, which eventually represents the record.

Embedding of each field's value (words) and field's name are obtained through learnable embedding matrix as follows.

$$Z_f[f_k^j] = [E_n[n^j]; E_v[v_k^j]]$$

where, $E_*[w]$ stands for embedding of w. $[;]$ denotes the concatenations. E_* are learnable parameters of size ($|v| \times d_{E_*}$), where $|v|$ is the size of vocabulary. Note that, here we use different embedding for both field's values and field' names. The $Z_f[f_k^j]$ denotes encoding for the k-th word in f^j field together with the field name. This Z_f is send to the horizontal connections of the field encoding.

Horizontal Connections in Field Encoding: Horizontal connections in field encoding are relating all fields words ($Z_f[f_*^j]$) inside of a record (r^j). Now, field words (f_*^j) can be either a sequence or bag-of-words (i.e. orderless/non-sequential). For example, in figure 1, the 'name' field contains two

words *'The'* and *'Punter'* as a sequence. However, if the 'price range' field contains two words—*'high'* and *'medium'*, to denote a restaurant offers foods of both high and medium price range, then these two words behave as bag-of-words.

To appease both sequence and bag-of-words nature together we build horizontal connection of field encoding in a distinct way. For sequence data we use Bi-LSTM (Graves et al., 2013) networks; for orderless bag-of-words we skip (with help of skip-interconnections (He et al., 2016)) this Bi-LSTM network.

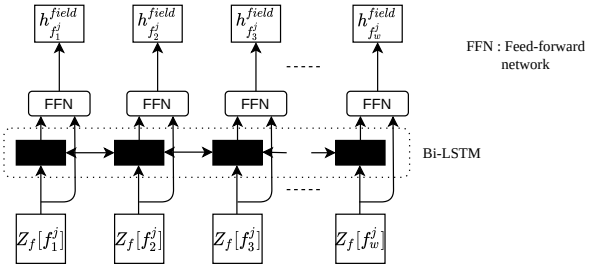


Figure 2: Horizontal connection in field encoding (for fields inside j-th records).

Eventually, a feed-forward network (equation 1) is used to merge skip-interconnections and Bi-LSTM. These skip-interconnections play an important role in our model while handling orderless/non-sequential data—we empirically show this in the experimental section 4.6. Figure 2 show such horizontal connections of field encoding.

The Bi-LSTM output for the field in j-th record is as follows,

$$h_{fj} = \text{BiLSTM}(Z_f[f_1^j], ..., Z_f[f_w^j])$$

Finally, we make use of an affine transformation on $[h_{f_k^j}; f_k^j]$.

$$h_{f_k^j}^{field} = \tanh(\mathbf{W_{fa}}[h_{f_k^j}; f_k^j] + \mathbf{b_{fa}}) \quad (1)$$

where, $\mathbf{W_{fa}}$ and $\mathbf{b_{fa}}$ are the learnable parameters. So, h_{fj}^{field} (and $h_{f_k^j}^{field}$ is k-th field) is output of horizontal connections of field encoding.

Hierarchical Connections in Field Encoding: The hierarchical connections in field encoding aim to accumulate all fields information ($h_{f_k^j}^{field}$) inside of a record (r_j) and gives a representation of the record from its fields point of view.

$$h_j^{mp} = \text{maxpool}(h_{fj}^{field})$$

For hierarchical connections in field encoding, we use max-pooling and key-query based self-attention mechanism (Conneau et al., 2017). The max-pooling is used because of our intuition that max-pooling help in capturing the essence of a record from its fields ($h_{f_k^j}^{field}$). We draw this intuition from popular convolution neural networks (Krizhevsky et al., 2012).
We find that use of max-pooling is more effective than using the last state of underlying horizontal connection. Remember, max-poling considers all states of underlying horizontal connection—which is helpful for long sequences.

For the key-query based self-attention on field encoding, we choose field values ($h_{f_*^j}^{field}$) as keys and max-pooled record value (h_j^{mp}) as query. Based on our previous intuition behind h_j^{mp}, the query of the self attentions holds essence of record (r_j).

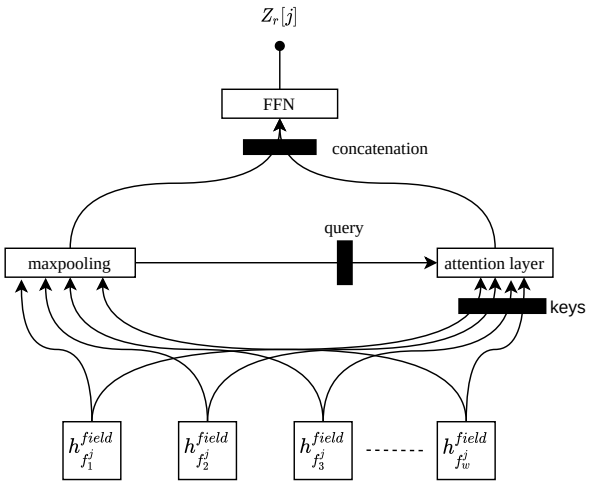


Figure 3: Hierarchical connections in field encoding (for fields inside j-th records).

For attention scoring, we use popular concatenative attention method (Luong et al., 2015).

$$score_{ji}^f = v_f^T \tanh(\mathbf{W_{fb}}[h_j^{mp}; h_{f_i^j}^{field}])$$
$$\alpha_{ji}^f = \text{softmax}(score_{ji}^f)$$
$$c_j^f = \sum_{i=1}^{m} \alpha_{ji}^f . h_{f_i^j}^{field}$$

here, $score^f \in \mathcal{R}$ denotes attention score, α^f denotes the attention weight, and c_j^f is the attention/context vector for r_j record.

At the end of hierarchical connections in field encoding, we represent each record (r_j) through

an affine transformation over concatenation of corresponding max-pooled value (h_j^{mp}) and context vector value (c_j^f).

$$Z_r[j] = \mathbf{W_{fc}}[h_j^{mp}; c_j^f] + \mathbf{b_{fc}}$$

Hence, $Z_r[j]$ is the output of field encoding for j-th records. Figure 3 shows the hierarchical connections of field encoding. It is worth to note here that as self-attention and max-pool operations don't rely on order of a data, hence there is no question of order-sensitiveness of hierarchical connections.

3.1.2 Record Encoding

The objective of record encoding is to find a vector representation for input data x in terms of its underlying record structures. The record encoding is quite similar to the field encoding. It contains both horizontal and hierarchical connections, which are same as field encoding. The horizontal connection output of record encoding is h_j^{record} for the j-th record. The final output of the record encoding is encoded representation $Z_d[x]$ of input data x.

3.2 Decoder

The decoder module consists of two key parts—structure-wise (i.e. record and field) attention gates and a copy module. Figure 4 shows a schematic view of our decoder.

3.2.1 State Update

The initial state of the decoder (s_0) is initialized through the output of record encoding, $Z_d[x]$. For updating the t-th step state (s_t) in the decoder, we use dual attention mechanisms (Liu et al., 2018; Gong et al., 2019) to merge attentions over record and field structures. We define attention over both record and field encoding on their horizontal connections outputs.

$$\beta_{tj}^r = \text{attention}(s_t, r_j)$$
$$\phi_t^r = \sum_{i=1}^{n} \beta_{ti}^r . h_{r_i}^{record}$$

where, β^r stands for attention weight (for j-th record, r_j, t-timestep of decoding) and ϕ_t^r is the attention/context vector of record encoding. Similarly, for field we get attention weight β^f.

Now, due to dual attention mechanism we modify effective attention weight of field encoding though $\gamma^f = \beta^f \times \beta^r$. Hence, the context vector (ϕ_t^f) of effective field attention is defined through γ^f.

Attention Gates: We introduce two attention gates for both field and record structures which help us to read context vectors ϕ_t^r and ϕ_t^f. We define these gates through current decoder state (s_t) and encoded data ($Z_d[x]$) as follows,

$$g_t^r = \sigma(\mathbf{W_{rg}} s_t + \mathbf{U_{rg}} Z_d[x] + \mathbf{b_{rg}})$$
$$g_t^f = \sigma(\mathbf{W_{fg}} s_t + \mathbf{U_{fg}} Z_d[x] + \mathbf{b_{fg}})$$

where, g_t^f and g_t^r are the attention gates for field and record context. Those two gates perform crucial function in our model as they handle content selection from the context vectors (which is nothing but encoded input) to decoder. The values of these gates change time to time to decide whether to inhibit (by value '0') and exhibit (by value '1') the content of context vectors. The attention context information is defined as below.

$$\hat{\phi}_t^r = g_t^r \odot \phi_t^r$$
$$\hat{\phi}_t^f = g_t^f \odot \phi_t^f$$

Finally, we update the decoder state with $\hat{c}_t^{record}$ and $\hat{c}_t^{field}$ as given below.

$$\tilde{s}_t = \tanh(\mathbf{W_d}[s_t; \hat{\phi}_t^{record}; \hat{\phi}_t^{field}])$$

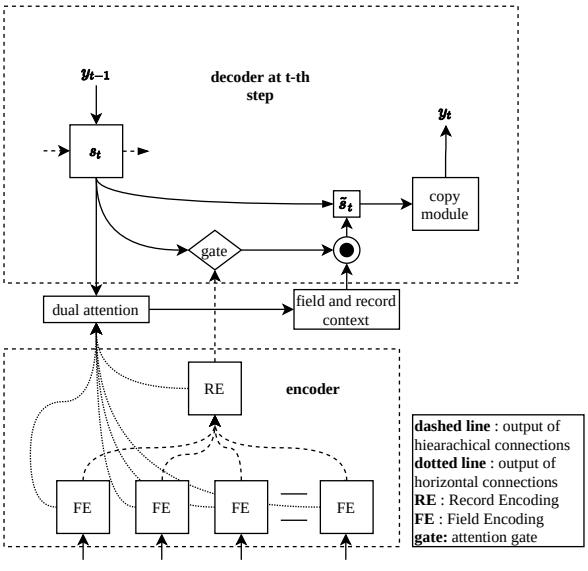


Figure 4: Proposed encoder-decoder model.

3.3 Copy Module

To handle rare words, we follows Gulcehre et al. (2016)'s conditional-copy techniques for our model, where the binary copy-indicator variable

(cp_t) in conditional copy-module defined as follows,

$$p(cp_t = 0|\tilde{s}_t, Z_d[x]) = \sigma(\mathbf{W_{cp}}[\tilde{s}_t; Z_d[x]])$$

We use input encoding $Z_d[x]$ with current decoder attention state $\tilde{s}_t$, to make the cp_t input-sensitive.

3.4 Loss Function

We use the negative log-likelihood function as loss function of our model:

$$loss = -\sum_{i=1}^{m} log(p(y_t^{(i)}|x^{(i)}, y_{<t}^{(i)})).$$

Two things are important to note here, (i) we use a $\langle$unk$\rangle$ whenever an out-of-vocabulary word appears in field; (ii) we never share embeddings between field's name and field's value.

4 Experiments

The experiment considers two popular benchmark datasets for data-to-text generations—WikiBio dataset and E2E dataset.

4.1 Baselines and Metrics

The following three baseline systems are considered in our experiment.

1. **Baseline 1:** It is a vanilla seq2seq model, based on the popular seq2seq (Sutskever et al., 2014) architecture and concatenative attention mechanism (Luong et al., 2015).

2. **Baseline 2:** To investigate the role of attentions in our model, this baseline model is considered where all attentions (at the decoder part) are removed from our proposed system.

3. **Baseline 3:** This is standard transformer (Vaswani et al., 2017) architecture based encoder-decoder model. We train this baseline model in our experiments from scratch.

Beside those three baselines, we consider two recent proposed data-to-text generator systems in our experiments. According to our knowledge, those systems have achieved high-scored performance on both WikiBio dataset and E2E dataset, in terms of automatic evaluations.

1. **Liu et al. (2018):** Liu et al. (2018) considers an encoder-decoder architecture, consists of a field-gating encoder (which enables structure-aware properties) and a dual attention-based decoder. Unlike our proposed encoder, Liu et al. (2018) encodes both records and fields information with parallel gating structures.

2. **Nie et al. (2019):** Nie et al. (2019) proposed an encoder-decoder model for data-to-text generation with self-attention mechanisms (Vaswani et al., 2017), for handling both sequential and non-sequential data. However, unlike our proposed model's where we incorporate both hierarchical and horizontal connections, Nie et al. (2019) mainly considers hierarchical self-attentions.

To evaluate the quality of generated text, we use three popular metrics—BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and METEOR (Banerjee and Lavie, 2005)[1]. We also use a recent data-to-text generation evaluation metric, PARENT (Dhingra et al., 2019), which considers both input data and reference texts in evaluation unlike above all three metrics which consider only reference. Dhingra et al. (2019) have extensively shown that PARENT metric correlates with human judgements more accurately than other automatic evaluations metrics.

As several past studies (Belz and Reiter, 2006; Reiter, 2018; Chaganty et al., 2018; Novikova et al., 2017a) have found that automatic evaluation is not a reliable way to evaluate data-to-text generation models, we also perform human evaluation on generated texts from our system.

4.2 Parameters Settings

The dimension of both field word and field name embedding are set to 200. We use a separate embeddings for a field name as well as for field values (i.e. vocabulary words). Adam optimization techniques (Kingma and Ba, 2015) (with initial learning rate=0.0001, β_1=0.9, β_2=0.999) are used to train the model. The depth of all BiLSTM models set as 2. In all cases, dropout value is fixed to 0.5. Most of the hyperparameters are tuned on predefined validation sets. In generating text from decoder the beam-search technique with beam size 4 is used. For baseline 3, we use standard transformer with stack size of 6 in both decoder and

[1]For BLEU, ROUGE and Meteor, https://github.com/tuetschek/e2e-metrics

dataset	instances	total words	tokens/sentence	sentences/instance	references/instance
WikiBio	728K	400K	26.1	1	1
E2E	50.5K	5.2K	14.3	1.5	8.1

Table 2: Dataset statistics.

encoder. In WikiBio dataset, while choosing sizes for vocabulary(or word types) and field types we closely follow Lebret et al. (2016). In most of the cases, we fix batch size to 32/64 and we train our model at most 120000 steps. We use NVIDIA GeForce RTX 2080 Ti graphics card for our experiments.

4.3 WikiBio Dataset and Results

Lebret et al. (2016) introduced WikiBio dataset from biographical articles on Wikipedia. Table 2 shows statistics of WikiBio dataset. From Table 3, we observe that our model achieves better outcomes in terms all automatic evaluation metrics than baselines and those two recent reported best results. Some examples of generated texts of our proposed system are given in Appendix A (Table 10).

model	BLEU	ROUGE-L	METEOR	PARENT
baseline 1	0.338	0.418	0.271	0.463
baseline 2	0.348	0.445	0.262	0.455
baseline 3	0.381	0.486	0.340	0.407
Liu et al. (2018)	0.447	0.528	0.363	0.538
Nie et al. (2019)	0.450	0.522	0.371	0.527
proposed method	**0.465**	**0.566**	**0.397**	**0.540**

Table 3: Results from WikiBio dataset.

4.4 E2E Dataset and Results

Novikova et al. (2017b) introduced E2E dataset 2 on restaurant text domain. From the comparison results presented in table 4, it is quite clear that our model outperforms the baselines and other reported systems, almost in every cases except Liu et al. (2018) model performs a bit better ($\sim 1\%$ compared to our model) in terms of PARENT metric for E2E Dataset. Some samples of generated text are provided in Appendix A (Table 11).

model	BLEU	ROUGE-L	METEOR	PARENT
baseline 1	0.517	0.520	0.344	0.569
baseline 2	0.534	0.572	0.350	0.572
baseline 3	0.568	0.594	0.425	0.642
Liu et al. (2018)	0.653	0.614	0.428	**0.726**
Nie et al. (2019)	0.662	0.612	0.399	0.663
proposed method	**0.675**	**0.683**	**0.442**	0.716

Table 4: Results on E2E dataset.

4.5 Analysis from Ablation Tests

To understand effectiveness of copy module, attention gates and encoder of our model, we do component ablation study based on BLEU and PARENT scores.

4.5.1 Ablation of Copy Module

Table 5 shows the copy module ablation results. From the reported BLEU and PARENT scores, it is quite clear that the copy module plays an important role in generating better text for both datasets.

	WikiBio		E2E	
	BLEU	PARENT	BLEU	PARENT
with copy module	0.465	0.540	0.675	0.716
without copy module	0.424	0.527	0.619	0.682

Table 5: Ablation of copy module (based on BLEU and PARENT score).

4.5.2 Ablation of Attention Gates

To observe the effectiveness of the attention gates we perform ablation tests on them. In terms of BLEU score, we find very small to no improvement. However, attention gates show a clear improvement in terms of PARENT metrics scores. Moreover, while doing qualitative analysis, we observe that the quality of generated texts is improved through these attention gates. Table 6 shows such qualitative analysis results. It may be noted that our model makes a few mistakes irrespective of whether attention gates are used or not. However, in terms of quality of generated text attention gates play an affirmative role as the number of wrongly inserted words is less for the model with attention gates compared to the model without gates.

	WikiBio		E2E	
	BLEU	PARENT	BLEU	PARENT
with attention gate	0.465	0.540	0.674	0.716
without attention gates	0.458	0.513	0.680	0.694

Table 7: Ablation of attention gates (based on BLEU and PARENT score).

WikiBio	
input	name[myles wilder], birth-date[january 28, 1933], birth-place[new york city, new york], death_date[april 20, 2010 -lrb- age 77-rrb-], death-place [temecula, california], occupation [television writer and producer], spouse [bobbe wilder -lrb- survives him -rrb-], article-title [myles wilder]
reference	myles wilder -lrb- january 28 , 1933 - april 20 , 2010 -rrb- was a television comedy writer and producer .
without attention gates	myles wilder -lrb- 28 , 1933 april , -rrb- held a television comedy writer and .
with attention gates	myles wilder -lrb- january 28 , 1933 – april 20 , 2010 -rrb- was an american television writer and producer .
E2E	
input	name[Blue Spice], eatType[restaurant], food[English], area[riverside], familyFriendly[yes], near[Rainbow Vegetarian Cafe]
one reference	there is a restaurant that provides food and is children friendly, near rainbow vegetarian cafe and the riverside and is called blue spice.
without attention gates	there is a soup restaurant that provides food and good spice. friendly, near rainbow vegetarian cafe and the riverside and blue.
with attention gates	near the rainbow vegetarian café in the riverside area is a restaurant called blue spice that serves english food and is children friendly.

Table 6: A qualitative analysis for the role of attention gates in our model (wrong word, word is not available in input) [E2E dataset contains several gold references for a single input data, but due to space constraint only one reference is given.].

	WikiBio		E2E	
	BLEU	PARENT	BLEU	PARENT
with all connections	0.465	0.540	0.675	0.716
without horizontal connections	0.369	0.483	0.532	0.580
without hierarchical connections	0.414	0.498	0.581	0.673

Table 8: Ablation of encoder connections (based on BLEU and PARENT score).

model	WikiBio		E2E	
	BLEU	PARENT	BLEU	PARENT
with skip-interconnection	0.421 ± 0.011	0.493±0.017	0.637 ± 0.009	0.682 ± 0.013
without skip-connection	0.413 ± 0.024	0.490±0.044	0.628 ± 0.021	0.652 ± 0.036

Table 9: Ablation test of skip-interconnections with shuffling records in input data.

4.5.3 Ablation of Encoder Connections

Through ablation test, we analyze the effectiveness of our encoder connections—both horizontal connections and hierarchical connections. Table 8 reports results of ablation test on encoder's connections. It is observed that the proposed model performed better when both connections present.

4.6 Analysis of Model with Shuffled Input

Earlier, we have mentioned that in order to encode both sequential and non-sequential (orderless) data through our proposed encoder, we introduced skip-interconnection to effectively handle them. To be more precise horizontal connections are responsible for the sequential data encoding, whereas hierarchical connections play essential roles for the non-sequential data. Finally, the skip-interconnections use both outputs from horizontal and hierarchical connections to nullify model bias toward record/field orders. In this section, we will investigate the role of skip-interconnections with random shuffling of records of input data. The aim of this experiment is to show the effectiveness of the proposed encoder on shuffled data. This experiment is evaluated through both intrinsic and extrinsic way.

Extrinsic Evaluation: Here we conduct ablation test on skip-interconnections with shuffling records in input data on both of the datasets. On each dataset's test set, such record shuffling are performed for five times . Table 9 presents effective of proposed encoder's skip-interconnections in terms of low fluctuation (standard deviations) measures on both PARENT and BLEU metric.

Intrinsic Evaluation: To more closely observe the effect of skip-interconnections on our model in handling shuffled input data, we show t-SNE plots (Maaten and Hinton, 2008) for encoded representations of input data with our encoder. Two random data instances are sampled from each of the two datasets (WikiBio and E2E), and each data instance is shuffled close to 30 different arrangements. We show t-SNE plots of encoded repre-

sentations of those shuffle data through our encoder. The well disentangled encoded representations of shuffled data (in figure 5) with skip-interconnections clearly prove effectiveness of skip-interconnections.

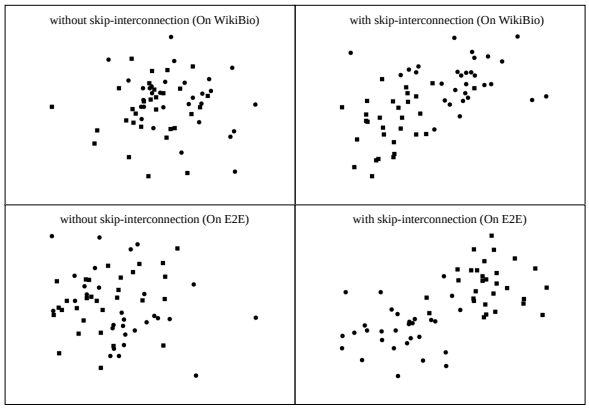


Figure 5: t-SNE plots for encoder representations on shuffled input data (circled and squared points represent two data different data instances).

4.7 Human Evaluation

In human-based evaluation for annotations purpose, we select four university graduate students from various majors. A sample of 150 generated texts from the proposed model is chosen for each of the two datasets (E2E and WikiBio) for the annotation task. Along with the generated text, we also provide input data and reference text to annotators. Every instance is annotated by at least three human annotators. In human-based evaluation, we primarily look for two essential qualities in generated texts–*adequacy* (or correctness) and *readability* (or fluency) (Gatt and Krahmer, 2018). The *adequacy* indicates whether appropriate/correct content of input data is contained within the generated text or not. The term *readability* defines fluency, clarity, and linguistic quality of a generated text. For adequacy and readability, every annotator is asked to rate each text on a scale of 1 to 5, where 5 is the best. Human evaluation results are presented in Figure 6 along with the inter-annotators agreement in terms of Krippendorff's alpha coefficient[2]. Evaluation results show that experiment on WikiBio dataset resulted in better readability and informativeness compared to the results obtained for E2E dataset.

[2]https://www.nltk.org/api/nltk.metrics.html

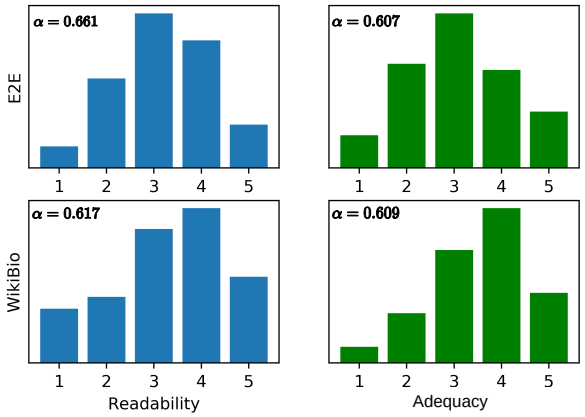


Figure 6: Average human rating of texts generated from the proposed model (**top left**:measures readability in E2E, **top right**: informativeness in E2E, **bottom left**: readability in WikiBio, **bottom right**: informativeness in WikiBio). α denotes the Krippendorff's alpha coefficient.

5 Related Works

The research presented in this paper is related to the recent data-driven data-to-text generation effort where text is generated from structured data (Angeli et al., 2010; Mei et al., 2016; Lebret et al., 2016; Liu et al., 2018). There are several types of data-driven text generation systems. Belz (2008) used probabilistic context-free grammar for text generation from structured data. Chen and Mooney (2008) introduced a strategic text generation technique for sportscasting of a simulated soccer game. Among data-driven text generations, Angeli et al. (2010) was probably the first to propose a domain-independent approach with an application on weather forecasting. With the advent of recurrent neural network language model (Mikolov et al., 2010), neural text generation models are proposed several in number and successfully applied to several different text generation tasks, from poem generation (Zhang and Lapata, 2014) to image captioning (Xu et al., 2015; Kiros et al., 2014; Karpathy and Fei-Fei, 2015). After the seq2seq model (Sutskever et al., 2014) and various attention mechanisms (Xu et al., 2015; Luong et al., 2015) are reported in the literature, the encoder-decoder model in neural text generation become quite ubiquitous. For selective generation task where readers focus on a certain selective part of the input, Mei et al. (2016) proposed an encoder-decoder model with an attention mechanism. In a concept-to-text generation where aim lies in generating text descriptions from complex concepts, the

encoder-decoder based models also achieve high accuracy (Lebret et al., 2016; Sha et al., 2018; Liu et al., 2018; Nema et al., 2018). For the dialogue system too, this kind of data-driven approach finds some important results (Wen et al., 2015; Shang et al., 2015). The encoder-decoder model has also shown promising results on table-to-text generation task (Bao et al., 2018; Gong et al., 2019).

6 Conclusion

In this paper, we propose an effective structural encoder for encoder-decoder based data-to-text generation, which carefully encodes record-field structured input data. With extensive experiments, we show that the proposed model is capable of handling both sequential and order-less (non-sequential) data. For selecting appropriate contents from encoded data, we incorporated attention gates in the proposed model. Evaluation of the model on WikiBio and E2E dataset brings out the potential of the proposed system in generating quality text. The immediate extension of this study may consider analysis of the model's behavior at sub-task levels, i.e., its effect on content selection, text/discourse planning, or on surface realization. These experiments may unveil more interesting features of the proposed model. Moreover, further research is needed to improve the quality of output text.

Acknowledgement

This work is partially supported by Science and Engineering Research Board (SERB), Dept. of Science and Technology (DST), Govt. of India through Grant File No. SPR/2020/000495.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

Anja Belz and Ehud Reiter. 2006. Comparing automatic and human evaluation of nlg systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Arun Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debiasing automatic metrics in natural language evalaution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653.

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Heng Gong, Xiaocheng Feng, Bing Qin, and Ting Liu. 2019. Table-to-text generation with effective hierarchical encoder on three dimensions (row, column and time). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3143–3152, Hong Kong, China. Association for Computational Linguistics.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 140–149, Berlin, Germany. Association for Computational Linguistics.

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

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

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

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1203–1213. The Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421. The Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 720–730. The Association for Computational Linguistics.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, and Mitesh M Khapra. 2018. Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1539–1550.

Feng Nie, Jinpeng Wang, Rong Pan, and Chin-Yew Lin. 2019. An encoder with non-sequential dependency for neural data-to-text generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 141–146.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017a. Why we need new evaluation metrics for nlg. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252.

Jekaterina Novikova, Ondrej Dusek, and Verena Rieser. 2017b. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pages 201–206. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ehud Reiter. 2018. A structured review of the validity of bleu. *Computational Linguistics*, 44(3):393–401.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the*

Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, pages 1577–1586. The Association for Computer Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

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

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Peihao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1711–1721. The Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

A Sample of Generated Texts from The Proposed Data-to-Text Generation Model on WikiBio and E2E Datasets

A Sample of generated texts on WikiBio dataset	
Instance 1	
input	name[pietro micheletti], birthdate[19 october 1900], deathdate[25 march 2005] birthplace[maciano di pennabilli, italy], deathplace[maciano di pennabilli, italy] allegiance[italy italy], branch[italian army], serviceyears[1917-1925], rank[major], battles[world war i fiume's war], laterwork[farmer manager], award[" military cross " " ordine di vittorio veneto" " knight of the order of merit of the italian republic "], articletitle[pietro micheletti]
reference	pietro micheletti -lrb- 19 october 1900 – 25 march 2005 -rrb- was an italian military commander .
generated text	pietro micheletti -lrb- october 19 , 1900 – march 25 , 2005 -rrb- was a italian army manager .
Instance 2	
input	name[jason buxton], nationality [canadian], occupation[film director, screenwriter] knownfor["blackbird"], articletitle[jason buxton]
reference	jason buxton is a canadian film director and screenwriter .
generated text	jason buxton is an canadian film director and screenwriter .
Instance 3	
input	name[bert crossthwaite], fullname[herbert crossthwaite], birthdate[4 april 1887], birthplace[preston, england], deathdate[20 may 1944], deathplace[birmingham, england], position[goalkeeper], years[–1906–1907 1907–1909 -1909 –1910 1910 – 1914 1914 –1915], clubs [blackpool fulham exeter city birmingham stoke], caps[0 1 2 49 0], goals[0 0 0 0 0], articletitle[bert crossthwaite]
reference	herbert crossthwaite -lrb- 4 april 1887 – 20 may 1944 -rrb- was an english footballer who played as a goalkeeper .
generated text	herbert bert crossthwaite -lrb- 4 april 1887 – 20 may 1944 -rrb- was an english footballer who played as a goalkeeper .

Table 10: A sample of generated texts from the proposed model on WikiBio dataset (wrong words generation, missing information)

A Sample of generated texts on E2E dataset	
Instance 1	
input	name[Giraffe], eatType[restaurant], food[French], area[city centre], familyFriendly[yes], near[Raja Indian Cuisine]
reference 1	the giraffe is a restaurant - family-friendly serving french food in the city centre, near the raja indian Cuisine
reference 2	giraffe, a family-friendly restaurant in the city centre serving french food is located near raja indian cuisine.
reference 3	giraffe is a restaurant located in the city centre near the raja indian cuisine. it offers french cuisine with a family-friendly atmosphere.
reference 4	city centre based restaurant near raja indian cuisine, giraffe is family-friendly and serves french cuisine.
reference 5	giraffe is a restaurant that offers french food with a family-friendly atmosphere. it is located in the city centre near the raja indian cuisine.
generated text	giraffe is a french restaurant in the riverside area near raja indian cuisine. it is not family-friendly.
Instance 2	
input	name[Cocum], eatType[pub], near[The Sorrento]
reference 1	cocum is a pub by the sorrento.
generated text	cocum is a pub near the ranch.
Instance 3	
input	name[Giraffe], eatType[pub], near[The Bakers]
reference 1	giraffe is a pub found near the bakers.
reference 2	a pub named giraffe is near the bakers.
generated text	giraffe is a pub near the bakers.

Table 11: A sample of generated texts from the proposed model on E2E dataset (wrong words generation, missing information)

Attention Is Indeed All You Need: Semantically Attention-Guided Decoding for Data-to-Text NLG

Juraj Juraska and **Marilyn Walker**
Natural Language and Dialogue Systems Lab
University of California, Santa Cruz
{jjuraska,mawalker}@ucsc.edu

Abstract

Ever since neural models were adopted in data-to-text language generation, they have invariably been reliant on extrinsic components to improve their semantic accuracy, because the models normally do not exhibit the ability to generate text that reliably mentions all of the information provided in the input. In this paper, we propose a novel decoding method that extracts interpretable information from encoder-decoder models' cross-attention, and uses it to infer which attributes are mentioned in the generated text, which is subsequently used to rescore beam hypotheses. Using this decoding method with T5 and BART, we show on three datasets its ability to dramatically reduce semantic errors in the generated outputs, while maintaining their state-of-the-art quality.

1 Introduction

Task-oriented dialogue systems require high semantic fidelity of the generated responses in order to correctly track what information has been exchanged with the user. Therefore, their natural language generation (NLG) components are typically conditioned on structured input data, performing *data-to-text* generation. To achieve high semantic accuracy, neural models for data-to-text NLG have invariably been reliant on extrinsic components or methods. While large pretrained generative language models (LMs), such as GPT-2 or T5, perform better in this respect, even they do not normally generate text that reliably mentions all the information provided in the input.

In this work, we study the behavior of attention in large pretrained LMs fine-tuned for data-to-text NLG tasks. We show that *encoder-decoder* models equipped with *cross-attention* (i.e., an attention mechanism in the decoder looking back at the encoder's outputs) are, in fact, aware of the semantic constraints, yet standard decoding methods do not fully utilize the model's knowledge. The method we propose extracts interpretable information from the model's cross-attention mechanism at each decoding step, and uses it to infer which slots have been correctly realized in the output. Coupled with beam search, we use the inferred slot realizations to rescore the beam hypotheses, preferring those with the fewest missing or incorrect slot mentions.

To summarize our contributions, the proposed semantic attention-guided decoding method, or SEA-GUIDE for short: **(1)** drastically reduces semantic errors in the generated text (shown on the E2E, ViGGO, and MultiWOZ datasets); **(2)** is domain- and model-independent for encoder-decoder architectures with cross-attention, as shown on different sizes of T5 and BART; **(3)** works out of the box, but is parameterizable, which allows for further optimization; **(4)** adds only a small performance overhead over beam search decoding; and **(5)** perhaps most importantly, requires no model modifications, no additional training data or data preprocessing (such as augmentation, segmentation, denoising, or alignment), and no manual annotation.[1]

2 Related Work

Several different approaches to enhancing semantic accuracy of neural end-to-end models have been proposed for data-to-text NLG over the years. The most common approach to ensuring semantic quality relies on over-generating and then reranking candidate outputs using criteria that the model was not explicitly optimized for in training. Reranking in sequence-to-sequence models is typically performed by creating an extensive set of rules, or by training a supplemental classifier, that indicates for each input slot whether it is present in the output utterance (Wen et al., 2015a; Dušek and Jurčíček,

[1]The code for SEA-GUIDE and heuristic semantic error evaluation can be found at https://github.com/jjuraska/data2text-nlg.

Proceedings of the 14th International Conference on Natural Language Generation (INLG), pages 416–431,
Aberdeen, Scotland, UK, 20-24 September 2021.

2016; Juraska et al., 2018; Agarwal et al., 2018; Kedzie and McKeown, 2020; Harkous et al., 2020).

Wen et al. (2015b) proposed an extension of the underlying LSTM cells of their sequence-to-sequence model to explicitly track, at each decoding step, the information mentioned so far. The coverage mechanism (Tu et al., 2016; Mi et al., 2016; See et al., 2017) penalizes the model for attending to the same parts of the input based on the cumulative attention distribution in the decoder. Chisholm et al. (2017) and Shen et al. (2019) both introduce different sequence-to-sequence model architectures that jointly learn to generate text and reconstruct the input facts. An iterative self-training process using data augmentation (Nie et al., 2019; Kedzie and McKeown, 2019) was shown to reduce semantic NLG errors on the E2E dataset (Novikova et al., 2017). Among the most recent efforts, the jointly-learned segmentation and alignment method of Shen et al. (2020) improves semantic accuracy while simultaneously increasing output diversity. Kedzie and McKeown (2020) use segmentation for data augmentation and automatic utterance planning, which leads to a reduction in semantic errors on both the E2E and ViGGO (Juraska et al., 2019) datasets.

In contrast to the above methods, our approach does not rely on model modifications, data augmentation, or manual annotation. Our method is novel in that it utilizes information that is already present in the model itself to perform semantic reranking.

Finally, related to our work is also controllable neural language generation, in which the constrained decoding strategy is often used, rescoring tokens at each decoding step based on a set of feature discriminators (Ghazvininejad et al., 2017; Baheti et al., 2018; Holtzman et al., 2018). Nevertheless, this method is typically used with unconditional generative LMs, and hence does not involve input-dependent constraints.

3 Semantic Attention-Guided Decoding

While we will evaluate the SEA-GUIDE method on ViGGO, E2E, and MultiWOZ, we develop the method by careful analysis of the cross-attention behavior of different pretrained generative LMs fine-tuned on the ViGGO dataset. ViGGO is a parallel corpus of structured meaning representations (MRs) and corresponding natural-language utterances in the video game domain. The MRs consist of a dialogue act (DA) and a list of slot-and-value pairs. The motivation for selecting ViGGO for developing the method was that it is the smallest dataset, but it provides a variety of DA and slot types (as shown in Table 1). The models used for the analysis were the smallest variants of T5 (Raffel et al., 2020) and BART (Lewis et al., 2020). We saved the larger variants of the models, as well as the other two datasets, for the evaluation.

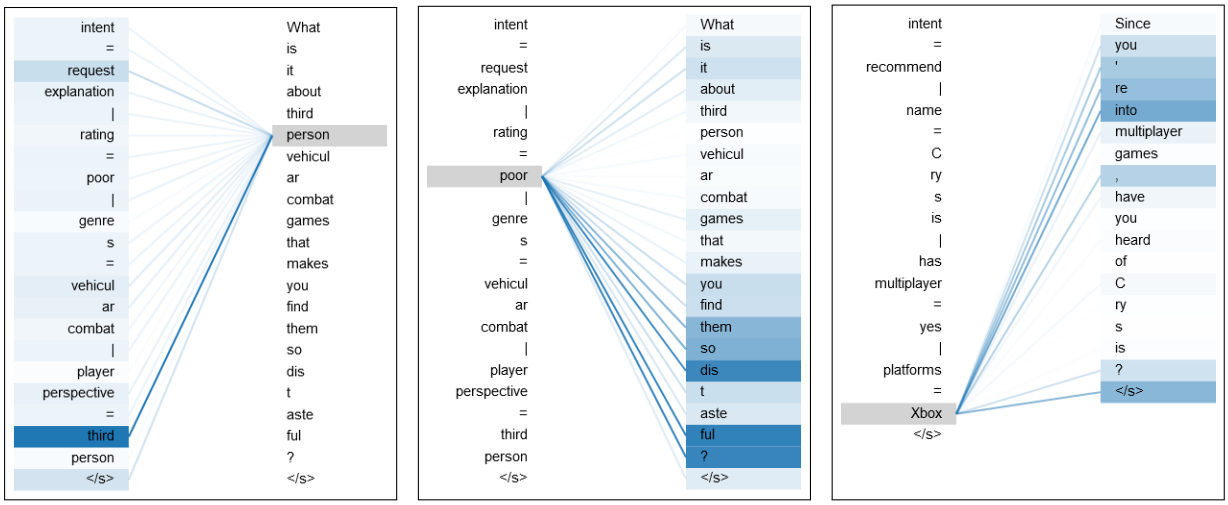
3.1 Interpreting Cross-Attention

Attention (Bahdanau et al., 2015; Luong et al., 2015) is a mechanism that was introduced in encoder-decoder models (Sutskever et al., 2014; Cho et al., 2014) to overcome the long-range dependencies problem of RNN-based models. It allows the decoder to effectively condition its output tokens on relevant parts of the encoder's output at each decoding step. The term *cross-attention* is primarily used when referring to the more recent transformer-based encoder-decoder models (Vaswani et al., 2017), to distinguish it from the *self-attention* layers present in both the encoder and the decoder transformer blocks. The cross-attention layer ultimately provides the decoder with a weight distribution at each step, indicating the importance of each input token in the current context.

Our results below will show that visualizing the attention weight distribution for individual cross-attention layers in the decoder – for many different inputs – reveals multiple universal patterns, whose combination can be exploited to track the presence, or lack thereof, of input slots in the output sequence. Despite the differences in the training objectives of T5 and BART, as well as their different sizes, we observe remarkably similar patterns in their respective cross-attention behavior. Below, we describe the three most essential patterns (illustrated in Figure 1) that we use in SEA-GUIDE.

3.1.1 Verbatim Slot Mention Pattern

The first pattern consistently occurs in the lowest attention layer, whose primary role appears to be to retrospectively keep track of a token in the input sequence that the decoder just generated in the previous step. Figure 1a shows an example of an extremely high attention weight on the input token "third" when the decoder is deciding which token to generate after "What is it about *third*" (which ends up being the token "person"). This pattern, which we refer to as the *verbatim* slot mention pattern, can be captured by maximizing the weight over all attention heads in the decoder's first layer.



(a) Verbatim slot mention (1st layer). (b) Paraphrased slot mention (3rd layer). (c) Unrealized slot mention (4th layer).

Figure 1: Visualization of cross-attention weight distribution for the 6-layer T5-small (trained on the ViGGO dataset) in 3 different scenarios. The left column in each corresponds to the input tokens, and the right to the tokens generated by the decoder. The darker the blue background shade, the greater the attention weight. Note that the weights are aggregated across all attention heads by extracting the maximum.

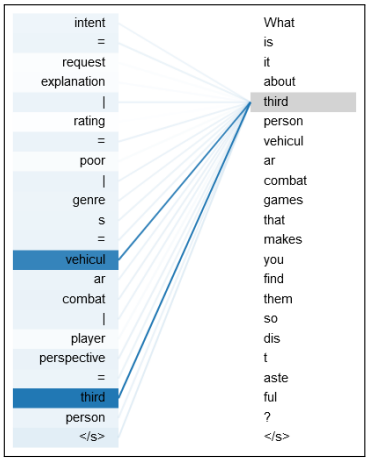


Figure 2: An example of the decoder paying equal attention (in the 5th layer of the 6-layer T5-small) to two slots in the input sequence when deciding what to generate next after "What is it about".

3.1.2 Paraphrased Slot Mention Pattern

Paraphrased slot mentions, on the other hand, are captured by the higher layers, at the moment when a corresponding token is about to be mentioned next. Essentially, as we move further up the layers, the cross-attention weights gradually shift towards input tokens that correspond to information that is most likely to *follow next* in the output, and capture increasingly more abstract concepts in general. Figure 1b shows an example of the RATING slot's value "poor" paraphrased in the generated utterance as "distasteful"; the first high attention value associated with the input token "poor" occurs when the decoder is about to generate the "dis" token.

At certain points during generation, however, the attention in the uppermost layers is distributed fairly evenly among multiple slots, because any of them could lead to a coherent continuation of the sentence. For example, the generated utterance in Figure 2 could have started with "What is it about vehicular combat games played from a third-person perspective that…", where the GENRES slot is output before the PLAYER PERSPECTIVE slot.

In order to recognize a paraphrased mention, without incorrectly capturing other slots considered, we propose averaging the cross-attention weights, using only the bottom half of the layers (e.g., layers 1 to 3 in the T5-small model).

3.1.3 Unrealized Slot Mention Pattern

The third pattern alleviates any undesired side effects of identifying paraphrased mentions using the second pattern, i.e., slots incorrectly assumed to be mentioned. Figure 1c illustrates an unrealized slot (PLATFORMS) being paid attention to in several decoding steps. The cross-attention weight distribution for the "Xbox" token in the 4th layer, shows that the decoder considered mentioning the slot at step 5 (e.g., "Since you're *an Xbox fan* and like multiplayer games,…"), as well as step 8 (e.g., "…into multiplayer games *on Xbox,*…"). The second pattern, depending on the sensitivity setting (see Section 3.2), might infer the PLATFORMS slot as a paraphrased mention at step 5 and/or 8.

However, the PLATFORMS slot's value is also paid attention to when the decoder is about to generate the EOS token and, importantly, without any

high attention weights associated with other slots at this step. This suggests that the model *is aware* that it omitted that slot. However, at that point, the decoder is more confident ending the sentence than realizing the missed slot after generating a question mark. This *unrealized* slot mention pattern is most likely to occur in the higher cross-attention layers, but not necessarily, so it is more effective to capture it by averaging the attention weights over all layers (at the last decoding step).

Note on Boolean Slots. With any of the three patterns described above, Boolean slots, such as HAS MULTIPLAYER in Figure 1c, typically have a high attention weight associated with their name rather than the value. This observation leads to a different treatment of Boolean slots, as described in Appendix B.1.

3.2 Slot Mention Tracking

We use the findings of the cross-attention analysis for automatic slot mention tracking in the decoder. During decoding, for each sequence, the attention weights associated with the next token to be generated are aggregated as per Section 3.1. Using configurable *thresholds*, the aggregated weights are then binarized, i.e., set to 1 if above the threshold, and 0 otherwise. This determines the sensitivity of the pattern recognition. Optionally, all but the maximum weight can be set to 0, in which case only a single input token will by implied even if the attention mass is spread evenly across multiple tokens. Finally, the indices of binarized weights of value 1, if any, are matched with their corresponding slots depending on which slot-span in the input sequence they fall into. For details on automatically extracting slot spans, see Appendix B.1.

3.2.1 Mention-Tracking Components

The three mention-tracking components, each of which operates on different attention layers and uses a different weight aggregation and binarization strategy, are summarized in Table 3. These components are executed in sequence and update one common slot-tracking object.

The first component, which tracks verbatim mentions, operates on the first attention layer only, with a high binarization threshold. Slot mentions identified by this component are regarded as high-confidence. The second component tracks paraphrased mentions, which are identified as slot mentions with low confidence, due to the partial ambiguity in mention detection using the second pattern (see Section 3.1.2). The third component only kicks in when the EOS token is the most probable next token. At that point, it identifies – with high sensitivity – slots that were not realized in the sequence (e.g., the PLATFORMS slot in Figure 1c), and removes the corresponding mention record(s). Only low-confidence mentions can be erased, while high-confidence ones are final once they are detected.

3.3 Semantic Reranking

Combining the slot mention tracking with beam search, for each input MR we obtain a pool of candidate utterances along with the semantic errors inferred at decoding time. We then rerank the candidates and pick the one with the fewest errors, resolving ties using the length-weighted log-probability scores determined during beam search.

4 Evaluation

In order to measure the proposed decoding method's performance in semantic error reduction, we first develop an *automatic* way of identifying erroneous slot mentions in generated utterances. In a human evaluation we establish that its performance is nearly perfect for all three datasets used for testing our models (see Section 4.1). We then use it to calculate the slot error rate (SER) automatically for all our model outputs across all datasets and configurations tested, which would be infeasible to have human annotators do.

Datasets. Besides ViGGO, which we use for fine-tuning the decoding (slot-tracking) parameters of the proposed SEA-GUIDE method, we evaluate its effectiveness for semantic error reduction on two *unseen* and *out-of-domain* datasets. While E2E (Novikova et al., 2017) is also a simple MR-to-text generation dataset (in the restaurant domain), MultiWOZ 2.1 (Eric et al., 2020) is a dialogic corpus covering several domains from which we extract system turns only, along with their MR annotations, along the lines of Peng et al. (2020) and Kale and Rastogi (2020). Table 1 gives an overview of the datasets' properties.

Setup. In our experiments, we fine-tune T5 and BART models of varying sizes on the above datasets' training partitions, select the best model checkpoints based on the BLEU score they achieve on the respective validation set, and evaluate them on the test sets while using different decoding meth-

	Size	Domains	DAs	Slots
ViGGO	6,900	1	9	14
E2E	51,426	1	1	8
MultiWOZ	70,530	7	13	27

Table 1: Dataset statistics, including the total number of dialogue act (DA) and slot types. For MultiWOZ, the numbers are calculated across system turns only.

	SER_{SA}	SER CI (95%)	Precision	IAA
ViGGO	2.77%	$2.19 \pm 1.55\%$	97.37%	1.00
E2E	3.98%	$3.91 \pm 1.73\%$	100%	1.00
MultiWOZ	1.19%	$1.35 \pm 0.91\%$	94.89%	0.90

Table 2: Human evaluation of the slot aligner's performance on each dataset. The IAA column indicates the Krippendorff's alpha reliability coefficient.

ods for inference. For beam search decoding, including when used as part of SEA-GUIDE, we use beam size 10 and early stopping, unless stated otherwise. All of our results are averaged over three runs with random initialization. For further details on training and inference parameters, we refer the reader to Appendix A.3.

4.1 Automatic Slot Error Evaluation

We evaluate our trained models performance with the standard NLG metrics BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007), ROUGE-L (Lin, 2004), and CIDEr (Vedantam et al., 2015), whose calculation is detailed in Appendix A.4. However, we also put substantial effort into developing a highly accurate heuristic *slot aligner* to calculate the semantic accuracy of generated utterances. The slot aligner is rule-based and took dozens of man-hours to develop, but it is robust and extensible to new domains, so it works on all three test datasets. Using the slot aligner, we count missed, incorrect, and repeated slot mentions, and determine the slot error rate (SER) as the percentage of these errors out of all slots.

To verify our slot aligner's performance, we take the generated utterances of one model per dataset for which it calculated a relatively high SER (indicated in the SER_{SA} column in Table 2). We then have one of the authors and an additional expert annotator manually label all of the errors as true or false positives. This corresponds to 38, 173 and 176 errors for ViGGO, E2E and MultiWOZ, respectively. From that we calculate the precision for each dataset, which turns out to be above 94% for each of the datasets. The almost perfect inter-annotator agreement (IAA), besides validating the precision, also suggests that the SER is an objective metric, and therefore well-suited for automation.

	Verbatim	Paraphrased	Unrealized
Layer agg.	1^{st} layer only	avg. over bottom half of layers	avg.
Head agg.	max.	max.	max.
Bin. threshold	0.9	0.4 (T5-small) 0.3 (BART-base)	0.1
Bin. max.	yes	no	no

Table 3: The final configuration of parameters used in each of the 3 mention-tracking components. The "Bin. max." row indicates whether only the maximum weight is kept during binarization, or all above the threshold.

Furthermore, we take samples of 72 ($\approx 20\%$), 63 ($\approx 10\%$) and 290 ($\approx 4\%$) of the generated utterances on ViGGO, E2E and MultiWOZ, respectively, annotate them for all types of errors, and calculate the *actual* SER confidence intervals (middle column). Their good alignment with the slot aligner SER scores, together with the high error classification precision, leads us to the conclusion that the slot aligner performs similarly to humans in identifying semantic errors on the above datasets.

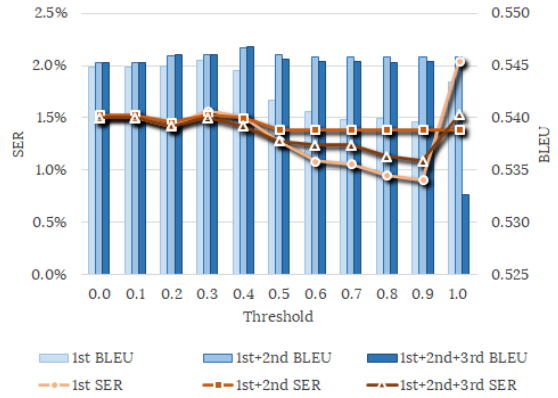
Besides SER evaluation, the slot aligner can also be used for beam reranking. Due to the handcrafted and domain-specific nature of the slot aligner, beam search with this reranking has a distinct advantage over SEA-GUIDE, which can be used for any domain out of the box. We therefore consider the results when using the slot-aligner reranking to be an upper bound for SEA-GUIDE in terms of SER.

4.2 SEA-GUIDE Parameter Tuning

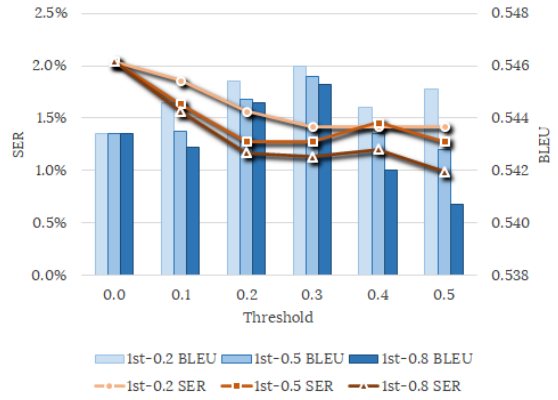
Each of the three mention-tracking components described in Section 3.2.1 has four configurable parameters, which we tuned by testing T5-small and BART-base, fine-tuned on the ViGGO dataset and equipped with SEA-GUIDE for inference. The parameter optimization was based on the insights obtained in Section 3.1 and a subsequent grid search, with results in Table 3.

For attention weight aggregation, we experimented with summing, averaging, maximizing, and normalizing. We determined *averaging* over layers and *maximizing* over heads to be the best combination for all three components. As for the binarization thresholds, Figure 3 shows the most relevant slice of the grid search space for each component, leading to the final threshold values.

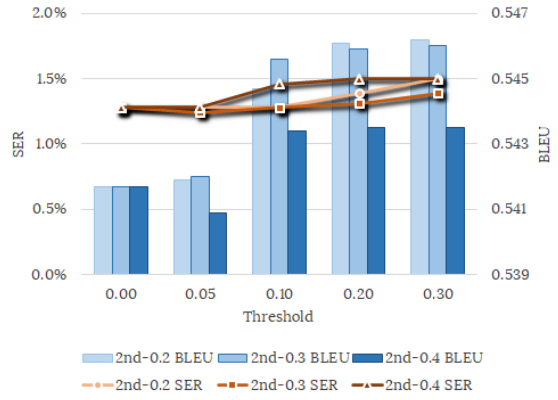
To show the effect of each slot-tracking component, we perform an ablation study with individual



(a) Threshold optimization for the 1st component (verbatim mentions), with the other components enabled or disabled. When enabled, the 2nd component's threshold was fixed at 0.3, and that of the 3rd at 0.1. Note that the threshold of 1.0 is equivalent to the 1st component being disabled, as attention weights are in the [0.0, 1.0] range.



(b) Threshold optimization for the 2nd component (paraphrased mentions), with the 1st component's threshold of 0.2, 0.5 and 0.8, and that of the 3rd component fixed at 0.1.



(c) Threshold optimization for the 3rd component (unrealized mentions), with the 2nd component's threshold of 0.2, 0.3 and 0.4, and that of the 1st component fixed at 0.5.

Figure 3: Effects of different parameter configurations of the 3 mention-tracking components on SER and BLEU of utterances generated by BART-base fine-tuned on ViGGO.

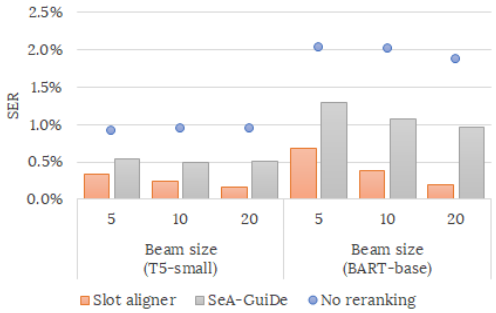


Figure 4: The effect of different beam size on the SER using different reranking methods on the ViGGO dataset. With greedy search decoding, the SER is 1.65% and 2.70% for T5 and BART, respectively.

components disabled.[2] As the plot in Figure 3a demonstrates, the 1st component by itself reduces the SER the most, but at the expense of the BLEU score, which decreases as the SER does – to the point where BLEU drops below 0.54 when the SER is at its lowest (0.91%), that is with a threshold of 0.9. For reference, the SER and the BLEU score achieved with beam search only are 2.04% and 0.543, respectively. Adding the 2nd component brings the BLEU score up to above 0.545, nevertheless the SER jumps to 1.39%. Finally, enabling the 3rd component too has a negligible negative effect on BLEU, but reduces the SER to 1.09%.

Figure 3b shows that the 2nd component gives optimal performance when its threshold is set to around 0.3. This setting maximizes BLEU, while keeping SER low. Beyond 0.3 the BLEU score starts dropping fast, and with a threshold of greater than 0.5, the 2nd component has barely any effect anymore. Similarly, Figure 3c shows the threshold value of 0.1 to be optimal in the 3rd component, when optimizing for both metrics. Thresholds higher than 0.3 cut off almost all aggregated weights in this component, virtually disabling it.

4.3 Effects of Beam Size on SEA-GUIDE

Since SEA-GUIDE uses beam search to generate the pool of candidates that it later reranks, we analyzed the effect of increasing the beam size on the SER of the final utterances. As Figure 4 shows for the ViGGO dataset, SEA-GUIDE certainly benefits from increasing the beam size from 5 to 10, but the benefit shrinks substantially (or disappears entirely, in case of T5-small) when further increased to 20. An analysis for the E2E dataset, with similar results, is presented in Appendix B.3.

[2]The 3rd component has no effect without the 2nd, so we do not consider the combination where only the 2nd is disabled.

5 Results

To maximize the performance of the models using SEA-GUIDE, the binarization thresholds (and possibly other parameters of the mention-tracking components) can be optimized for each model and dataset on the validation set. In our evaluation, however, we focused on demonstrating the effectiveness of this decoding method out of the box. That being said, even common decoding methods, such as simple beam search or nucleus sampling (Holtzman et al., 2019), usually benefit from parameter optimization (e.g., beam size, or the p-value) whenever used with a different model or dataset.

5.1 SEA-GUIDE Performance

While developing the SEA-GUIDE method we analyzed the behavior of cross-attention on both the T5-small and the BART-base model; interestingly, the decoding performs best for both with nearly the same configuration. The only difference is the 2nd component's binarization threshold (see Table 3), accounting for the fact that BART-base has 50% more attention heads than T5-small, which causes the attention weights to be more spread out.

The upper half of Table 4 compares the two models' performance with SEA-GUIDE vs. other decoding methods, as well as against three state-of-the-art baselines. As the results show, both models, when using SEA-GUIDE, significantly reduce the number of semantic errors in the generated outputs compared to using greedy search ($\approx$ 3.4 and 2.5 times in case of T5 and BART, respectively) or simple beam search ($\approx$ 1.9 times both). As expected, the slot-aligner (SA) reranking achieves even better results thanks to the handcrafted rules it relies on. In addition, the overall high automatic metric scores suggest that the fluency of utterances generated using SEA-GUIDE does not suffer.

Finally, compared to the baseline models, T5-small performs on par with the state-of-the-art DataTuner in terms of automatic metrics, yet maintains a 3.4-times lower SER. This corresponds approximately to K&M baseline's SER, whose automatic metrics, however, are significantly worse. BART-base outperforms T5-small according to most metrics, but its SER is more than double.

5.2 Cross-Model Robustness

In addition to T5-small and BART-base, we fine-tune a larger variant of each of the models, namely, T5-base and BART-large (see Appendix A.3 for model specifications), on the ViGGO dataset, and evaluate their inference performance when equipped with SEA-GUIDE. We do not perform any further tuning of the decoding parameters for these two models, only slightly lower the binarization thresholds (as we did for BART-base) to account for the models having more attention heads and layers. The thresholds we use for the 2nd and 3rd components are $\langle 0.3, 0.1 \rangle$ and $\langle 0.2, 0.05 \rangle$ for T5-base and BART-large, respectively.

The results in the lower half of Table 4 show that these two larger models, fine-tuned on ViGGO, benefit from SEA-GUIDE beyond just the effect of beam search. T5-base performs significantly better across the board than its smaller T5 variant, so there is less room for improvement to begin with. In fact, the SER using greedy search is so low (0.61%, in contrast to T5-small's 1.65%) that beam search causes it to increase. Nevertheless, SEA-GUIDE improves on both, while slightly boosting the other automatic metrics as well.

Model		BLEU	MET.	ROUGE	CIDEr	SER ↓
S2S		0.519	0.388	0.631	2.531	2.55%
DT		**0.536**	**0.394**	**0.640**	**2.700**	1.68%
K&M		0.485	0.380	0.592	2.454	***0.46%***
T5-small	GS	0.519	0.387	0.631	2.647	1.65%
	BS	0.540	0.392	0.636	2.685	0.95%
	SA	**0.541**	**0.393**	**0.637**	**2.695**	**0.24%**
	SG	**0.541**	**0.393**	**0.637**	**2.695**	0.49%
BART-base	GS	0.524	0.386	0.635	2.629	2.70%
	BS	0.544	0.393	**0.639**	2.679	2.02%
	SA	**0.547**	**0.394**	**0.639**	**2.704**	**0.39%**
	SG	0.545	0.393	**0.639**	2.698	1.07%
T5-base	GS	0.527	**0.394**	**0.639**	**2.682**	0.61%
	BS	0.534	**0.394**	0.636	2.664	0.66%
	SA	**0.536**	**0.394**	0.637	2.672	**0.19%**
	SG	**0.536**	**0.394**	0.637	2.670	0.46%
BART-large	GS	0.508	0.378	0.616	2.452	5.50%
	BS	0.535	0.391	0.628	2.612	1.78%
	SA	**0.538**	**0.394**	**0.631**	**2.659**	**0.27%**
	SG	0.533	0.391	0.627	2.613	1.41%

Table 4: Models tested on the ViGGO dataset using different decoding methods: greedy search (GS), beam search with no reranking (BS), beam search with slot-aligner reranking (SA), and SEA-GUIDE (SG). Baselines compared against are Slug2Slug (Juraska et al., 2019) (S2S), DataTuner (Harkous et al., 2020) (DT), and Kedzie and McKeown (2020) (K&M). The best results are highlighted in bold for each model. SER scores of baselines reported by the authors themselves, rather than calculated using our slot aligner, are highlighted in italics, and they do not correspond exactly to our SER results.

	Model	BLEU	MET.	ROUGE	CIDEr	SER ↓
	S2S	0.662	0.445	0.677	2.262	0.91%
	S_1^R	**0.686**	**0.453**	**0.708**	**2.370**	N/A
	K&M	0.663	**0.453**	0.693	2.308	***0.00%***
T5-small	GS	0.670	**0.454**	0.692	2.244	1.60%
	BS	0.667	0.453	**0.694**	**2.361**	2.85%
	SA	**0.675**	0.453	0.690	2.341	**0.02%**
	SG	0.675	0.453	0.690	2.340	0.04%
BART-base	GS	0.667	**0.454**	0.694	2.276	1.97%
	BS	0.670	**0.454**	**0.701**	**2.372**	3.39%
	SA	**0.680**	0.453	0.695	2.350	**0.02%**
	SG	**0.680**	0.453	0.695	2.347	0.08%
T5-base	GS	0.668	**0.459**	0.692	2.282	1.85%
	BS	0.667	0.453	**0.697**	**2.387**	3.94%
	SA	**0.682**	0.454	0.691	2.375	**0.03%**
	SG	**0.682**	0.454	0.691	2.374	0.05%

Table 5: Models tested on the E2E dataset, compared against the following baselines: Slug2Slug (Juraska et al., 2018), (S2S) S_1^R (Shen et al., 2019), and Kedzie and McKeown (2020) (K&M).

	Model	BLEU	$BLEU_R$	MET.	SER ↓	SER_E ↓
	SCG	N/A	0.308	N/A	***0.53%***	N/A
	K&R	N/A	**0.351**	N/A	N/A	1.27%
T5-small	GS	**0.367**	**0.351**	**0.325**	1.15%	1.36%
	BS	0.359	0.344	0.323	1.06%	1.19%
	SA	0.360	0.344	0.323	**0.41%**	**0.63%**
	SG	0.360	0.344	0.323	0.60%	0.85%
BART-base	GS	**0.372**	**0.356**	**0.326**	1.18%	1.17%
	BS	0.363	0.346	0.323	1.12%	1.02%
	SA	0.364	0.347	0.324	**0.40%**	**0.60%**
	SG	0.363	0.347	0.323	0.63%	0.72%

Table 6: Models tested on MultiWOZ, compared against the following baselines: SC-GPT (Peng et al., 2020) (SCG) and Kale and Rastogi (2020) (K&R).

The almost twice-as-large BART-large model performs rather poorly in our experiments, in fact, significantly underperforming its smaller variant.[3] We therefore refrain from drawing any conclusions for this model, although SEA-GUIDE offers a definite improvement in SER over simple beam search.

5.3 Domain Transferability

We achieve similar results when evaluating across domains. Table 5 shows that using SEA-GUIDE with all three models fine-tuned on E2E reduces the SER down to almost zero, with performance for the other metrics comparable to the state-of-the-art baseline.[4] In fact, SEA-GUIDE is nearly as effective at reducing errors in this dataset as the heuristic slot aligner (SA). Table 6 compares our models against two recent baselines on the MultiWOZ dataset, where the effectiveness of SEA-GUIDE on SER reduction is comparable to that on the ViGGO dataset. All in all, on both the E2E and the MultiWOZ dataset, our models equipped with SEA-GUIDE for inference perform similarly to the best baselines for both SER and the other metrics *at the same time*, whereas the baselines individually perform well according to one at the expense of the other.

5.4 Slot Error Detection Examples

Table 7 shows several utterances generated for corresponding input MRs in the video game domain, along with the errors SEA-GUIDE detected, if any. In the first example, all slots are correctly mentioned, and SEA-GUIDE agrees. This utterance was ultimately selected during reranking over the beam search's choice, "The Room is an excellent first person point-and-click puzzle game.", which has one of the genres omitted.

The second example again showcases a successful identification of all slot mentions by SEA-GUIDE, this time in an utterance where our heuristic slot aligner incorrectly found an error in the HAS_LINUX_RELEASE slot mention (presumably confused by the negation shortly before the keyword "Linux").

The third example shows that SEA-GUIDE is not confused by the difference between "excellent" (input slot value) and "amazing" (the corresponding slot realization), and it correctly identifies that the PLAYER_PERSPECTIVE slot is not mentioned.

In the fourth example, SEA-GUIDE fails to recognize the incorrect mention of the HAS_LINUX_RELEASE slot. This is the most common mistake our decoding method makes.

The final example does not contain any errors, yet the heuristic slot aligner was confused by the negation formulation (as "I wish it was") and identified the AVAILABLE_ON_STEAM slot as incorrect, while SEA-GUIDE did not. This demonstrates the robustness of SEA-GUIDE beyond exact matching. Yet despite the individual slots being mentioned semantically correctly, the generated utterance as a whole does not sound quite right. A better utterance

[3] We observed that it frequently misrepresents names, such as "Transportal Tycoon" instead of "Transport Tycoon", which we think may be the consequence of the extremely small size of the ViGGO training set relative to the model's size.

[4] We were unable to successfully train BART-large on E2E due to the memory limitations of our computational resources.

give_opinion(NAME[The Room], RATING[excellent], GENRES[indie, point-and-click, puzzle], PLAYER_PERSPECTIVE[first person])
The Room is an excellent indie point-and-click puzzle game. The first person perspective makes it easy to play. **Errors:** N/A
verify_attribute(name[Small World 2], RATING[poor], HAS_LINUX_RELEASE[yes])
You said earlier that you disliked Small World 2. Are you generally not fond of Linux games? **Errors:** N/A
request_explanation(RATING [excellent], GENRES [adventure, hack-and-slash, indie], PLAYER_PERSPECTIVE [third person])
What is it about indie hack-and-slash adventure games that you find so amazing? **Errors:** PLAYER_PERSPECTIVE
inform(NAME[RollerCoaster Tycoon], RATING[good], GENRES[simulation, strategy], PLATFORMS[Xbox, PC], AVAILABLE_ON_STEAM[no], HAS_LINUX_RELEASE[yes], HAS_MAC_RELEASE[no])
RollerCoaster Tycoon is a good simulation strategy game for Xbox and PC. It is not available on Steam, Linux, or Mac. **Errors:** N/A (incorrect)
give_opinion(NAME [Mirror's Edge Catalyst], RATING [poor], AVAILABLE_ON_STEAM [no])
Mirror's Edge Catalyst is a pretty bad game, I really wish it was available on Steam. **Errors:** N/A

Table 7: Examples of slot-mention tracking results using SEA-GUIDE. Each gray row is the input MR for the corresponding utterance in the subsequent row.

would be something along the lines of "Mirror's Edge Catalyst is a disappointment, I really wish they released it on Steam as well.".

All in all, SEA-GUIDE chooses semantically correct utterances that are fluent and adequate, except for the rare case like in the last example.

6 Discussion

In the previous section, we showed that SEA-GUIDE is highly effective at reducing semantic errors across different models and domains, and that without compromising on the generated utterances' fluency. On datasets other than E2E, it does not quite match the performance of beam search combined with our slot aligner-based reranking, but then again, the slot aligner is a hand-crafted tool with complex rules, requiring a good deal of domain knowledge, and suffering thus significantly in scalability. While these two decoding methods have a lot in common – both being based on beam search and subsequent candidate reranking – their difference lies in the identification of slot mentions; SEA-GUIDE identifies them *automatically* during the decoding, utilizing the model's cross-attention weights at each step, as opposed to relying on string-matching rules post decoding, which need to be extended for any new domains.

Despite working conveniently out of the box, SEA-GUIDE does not come with a computational overhead caveat. Performing inference on a GPU, SEA-GUIDE is a mere 11–18% slower than beam search with slot aligner-based reranking, while we observed no performance difference on a CPU (see Appendix B.4 for a detailed analysis).

6.1 Limitations of SEA-GUIDE

SEA-GUIDE's ability to recognize slot errors is limited to missing and incorrect slot mentions, which are the most common mistakes we observed models to make on the data-to-text generation task. Duplicate slot mentions are hard to identify reliably because the decoder inherently pays attention to certain input tokens at multiple non-consecutive steps (such as in the example in Figure 1b). And arbitrary hallucinations are entirely beyond the scope of this method, as there is no reason to expect cross-attention to be involved in producing input-unrelated content, at least not in a foreseeable way.

As we see in example #4 in Table 7, Boolean slots occasionally give SEA-GUIDE a hard time, as the decoder appears not to be paying a great deal of attention to Boolean slots' values throughout the entire decoding in many cases. We plan to investigate if the performance can be improved for Boolean slots, perhaps by modifying the input format or finding a more subtle slot mention pattern.

7 Conclusion

We presented a novel decoding method, SEA-GUIDE, that makes a better use of the cross-attention component of the already complex and enormous pretrained generative LMs to achieve significantly higher semantic accuracy for data-to-text NLG, while preserving the otherwise high quality of the output text. It is an automatic method, exploiting information already present in the model, but in an interpretable way. SEA-GUIDE requires no training, annotation, data augmentation, or model modifications, and can thus be effortlessly used with different models and domains.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback. This research was supported by NSF AI Institute Grant No. 1559735.

References

Shubham Agarwal, Marc Dymetman, and Eric Gaussier. 2018. Char2char generation with reranking for the e2e nlg challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 451–456.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Ashutosh Baheti, Alan Ritter, Jiwei Li, and William B Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3970–3980.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 633–642.

Kyunghyun Cho, Bart van Merriënboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands. Association for Computational Linguistics.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 422–428.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2410–2424.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649.

Juraj Juraska, Kevin Bowden, and Marilyn Walker. 2019. Viggo: A video game corpus for data-to-text generation in open-domain conversation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 164–172.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker. 2018. A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 152–162.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102.

Chris Kedzie and Kathleen McKeown. 2019. A good sample is hard to find: Noise injection sampling and self-training for neural language generation models. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 584–593, Tokyo, Japan. Association for Computational Linguistics.

Chris Kedzie and Kathleen McKeown. 2020. Controllable meaning representation to text generation: Linearization and data augmentation strategies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5160–5185.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of*

the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960.

Feng Nie, Jin-Ge Yao, Jinpeng Wang, Rong Pan, and Chin-Yew Lin. 2019. A simple recipe towards reducing hallucination in neural surface realisation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2673–2679.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 172–182.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4060–4067.

Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. 2020. Neural data-to-text generation via jointly learning the segmentation and correspondence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7155–7165.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 76–85.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010. Curran Associates Inc.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *SIGDIAL Conference*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*.

	Training	Validation	Test
ViGGO	5,103	714	1,083
E2E	42,063	4,672	4,693
MultiWOZ	55,951	7,286	7,293

Table 8: Overview of the dataset partitions.

A Appendix

A.1 Additional Dataset Details

Table 8 shows the number of examples in the training, validation and test partitions of all the datasets used in the evaluation of the SEA-GUIDE method.

A.2 Data Preprocessing

When preprocessing input meaning representations (MRs) before training a model or running inference, we first parse the dialogue act (DA) types, if present, and all slots and their values from the dataset-specific format into an intermediate list of slot-and-value pairs, keeping the original order. Although typically indicated in the MR differently from slots, we treat the DA type as any other slot (with the value being the DA type itself, and assigning it the name "intent").

Next, we rename any slots that do not have a natural-language name (e.g., "priceRange" to "price range", or "has_mac_release" to "has Mac release"). Slot values are left untouched. We do this to take advantage of pretrained language models' ability to model the context when the input contains familiar words, as opposed to feeding it code names with underscores and no spaces.

Finally, we convert the updated intermediate list of slots and their values to a string. The '|' symbol is used for separating slot-and-value pairs from each other, while the '=' is used within each pair to separate the value from the slot name. The result for an MR from ViGGO can look as follows:

```
intent = request_explanation
| rating = poor | genres =
vehicular_combat | player
perspective = third_person
```

A.3 Model and Training Parameters

The pretrained models that we fine-tuned for our experiments are the PyTorch implementations in the Hugging Face's Transformers[5] package. The models' sizes are indicated in Table 9.

We trained all models using a single Nvidia RTX 2070 GPU with 8 GB of memory and CUDA version 10.2. The training parameters too are summarized in Table 9. For all models, we used the AdamW optimizer with a linear decay after 100 warm-up steps. The maximum sequence length for both training and inference was set to 128 for ViGGO and E2E, and 160 for MultiWOZ.

	Layers	Heads	Hidden state size	Total parameters
T5-small	6+6	8	512	$\approx$ 60M
BART-base	6+6	12	768	$\approx$ 139M
T5-base	12+12	12	768	$\approx$ 220M
BART-large	12+12	16	1024	$\approx$ 406M

Table 9: Overview of the model specifications.

	Batch size	Learning rate	Epochs
T5-small	32/64/64	2×10^{-4}	20/20/30
BART-base	32/32/32	1×10^{-5}	20/20/25
T5-base	16/16/ –	3×10^{-5}	20/20/ –
BART-large	16/ – / –	4×10^{-6}	20/ – / –

Table 10: Overview of the training parameters used in our experiments. Batch size and the number of epochs are indicated per dataset (ViGGO/E2E/MultiWOZ).

A.4 Evaluation Metric Calculation

The four non-SER automatic metrics that we report in our results (i.e., BLEU, METEOR, ROUGE-L, and CIDEr) are calculated using the E2E evaluation script[6] developed for the E2E NLG Challenge (Dušek et al., 2018). We also verified that the single-reference BLEU score calculation in the E2E script corresponds to that in the SacreBLEU[7] Python package. As a result, BLEU scores calculated either way are directly comparable.

To ensure a fair comparison with the MultiWOZ baselines (Peng et al., 2020; Kale and Rastogi, 2020), we additionally report BLEU scores calculated using the RNNLG evaluation script[8], which their respective authors used in their own evaluation. We denote it BLEU_R in our result tables. Moreover, Kale and Rastogi (2020) calculated SER on utterance level, rather than slot level, and that using *exact* slot value matching in the utterance. We thus wrote a script to also perform this type of naive SER evaluation, in addition to our slot aligner-based SER evaluation. We report its results as SER_E.

[5] https://huggingface.co/transformers/

[6] https://github.com/tuetschek/e2e-metrics

[7] https://pypi.org/project/sacrebleu/

[8] https://github.com/shawnwun/RNNLG/

B Additional SEA-GUIDE Evaluation

B.1 Slot Mention Tracking Details

In order to be able to take advantage of the attention weight distribution patterns, the decoder needs to be aware of which input token span corresponds to which slot. To this end, we parse the input MRs on-the-fly – which is trivial given the structured nature of MRs – as each batch is being prepared for inference, and create a list of slot spans for each MR in the batch.[9] In fact, we indicate the spans for slot names and slot values separately, and for list-values down to individual list elements, for a higher specificity. Since Boolean slot mentions are tracked by their name rather than value, we also indicate for each slot whether it is Boolean or not. This information can be provided explicitly to the data loader, otherwise it is automatically inferred from the dataset's ontology based on all the possible values for each slot.

Note that, although our data preprocessing converts DA type indications in the MRs to the same format as slots (see any of the left columns in Figure 1), we exclude them from the slot-span lists, as they are not actual content slots to be tracked. Separator tokens (such as '|' or '=') present in the preprocessed MR are not included in the spans, and are, as a result, ignored during the slot mention tracking.

B.2 Parameter Tuning for T5

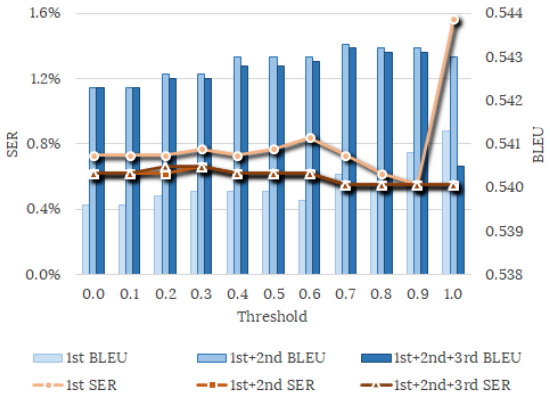
When optimizing the mention-tracking components' parameters for T5-small, we observe similar trends as with BART-base (see Figure 5). One difference is that enabling the 2nd component not only significantly increases the BLEU score, but also lowers the SER, while the 3rd component appears to only have a negligible effect (see Figure 5a).

B.3 Effects of Beam Size on E2E

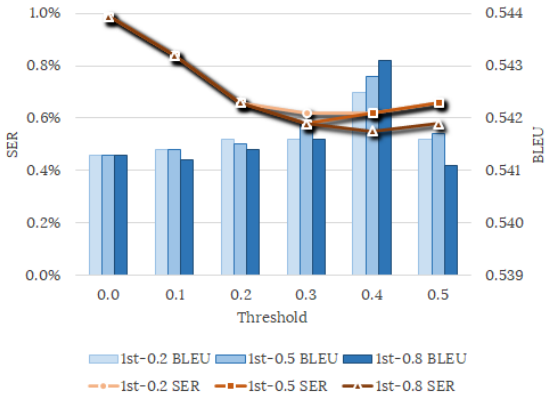
On the E2E dataset, decoding using SEA-GUIDE is even more effective in reducing SER than on ViGGO. Across all beam sizes, its performance is comparable to beam search with slot aligner reranking, and there is also only a limited gain from increasing the beam size to 20 (see Figure 6).

It is worth noting that, using beam search with no reranking, the SER dramatically increases with the increasing beam size. This is likely caused by the relatively heavy semantic noise in the E2E

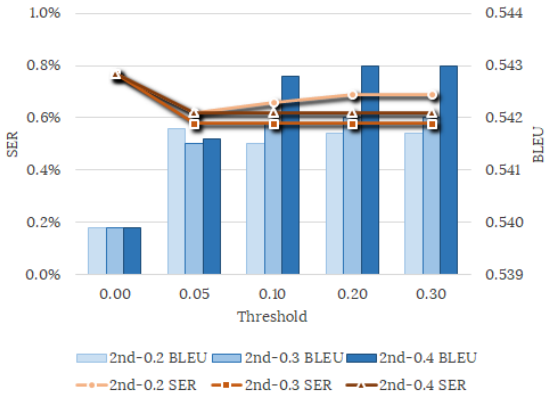
[9] This is done on token level, and the result varies thus from model to model depending on its tokenizer.



(a) Threshold optimization for the 1st component (verbatim mentions), with the other components enabled or disabled. When enabled, the 2nd component's threshold was fixed at 0.3, and that of the 3rd at 0.1. Note that the threshold of 1.0 is equivalent to the 1st component being disabled, as attention weights are in the [0.0, 1.0] range.



(b) Threshold optimization for the 2nd component (paraphrased mentions), with the 1st component's threshold of 0.2, 0.5 and 0.8, and that of the 3rd component fixed at 0.1.



(c) Threshold optimization for the 3rd component (unrealized mentions), with the 2nd component's threshold of 0.2, 0.3 and 0.4, and that of the 1st component fixed at 0.5.

Figure 5: Effects of different parameter configurations of the 3 mention-tracking components on SER and BLEU of utterances generated by T5-small fine-tuned on ViGGO.

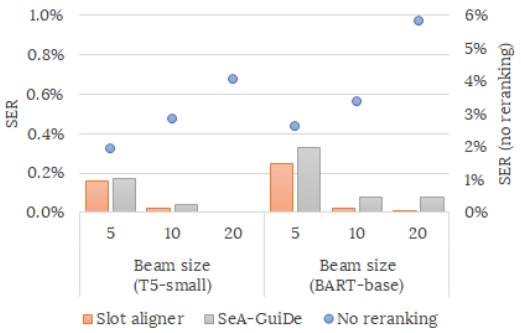


Figure 6: The effect of different beam size on the SER using different reranking methods on the E2E dataset. With greedy search decoding, the SER is 1.60% and 1.97% for T5 and BART, respectively.

training set, resulting in more slot errors in the generated utterances the less greedy the decoding is. Some form of semantic guidance is thus all the more important for the model in this scenario.

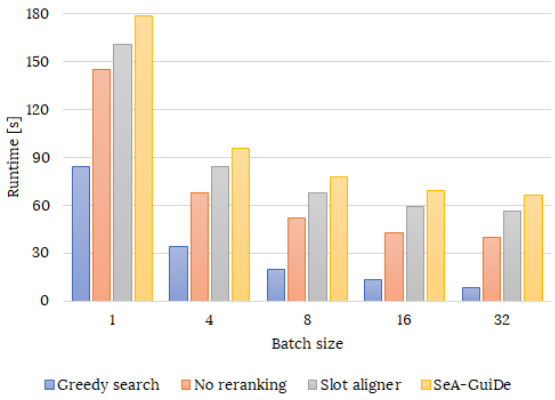
B.4 Inference Performance

In order to assess the computational overhead the SEA-GUIDE method introduces during inference, we measure the inference runtime of the T5-small model fine-tuned on ViGGO. For all beam search-based methods (including SEA-GUIDE), the beam size was set to 10, and early stopping was enabled.

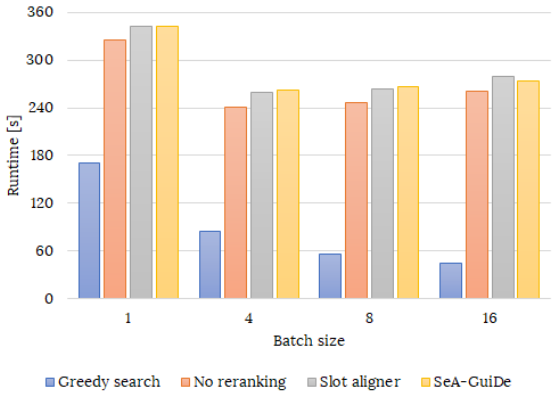
The results in Figure 7a show a distinct but expected overhead across all batch sizes when running inference on a GPU. The overall increase in runtime is 11–18% over beam search with slot aligner-based reranking, which is the method computationally most similar to SEA-GUIDE, as it too involves reranking on top of beam search. The slot aligner-based reranking itself adds a constant amount of 16 seconds on top of simple beam search, which corresponds to an 11-40% increase for the range of batch sizes in the plot.

When performing the same inference on a CPU, on the other hand, the overhead SEA-GUIDE introduces to beam search is no greater than that of the slot aligner-based reranking (see Figure 7b). This suggests that further optimization of SEA-GUIDE for GPU, especially by minimizing the communication between the GPU and the CPU during the decoding, could bring the overhead of SEA-GUIDE inference on a GPU down to the same level as that of the slot aligner-based reranking.

Considering the large improvement in semantic accuracy the SEA-GUIDE method delivers in the tested models, we deem the observed computational overhead reasonable and acceptable.



(a) Inference using a GPU (RTX 2070 with 8 GB of memory).



(b) Inference using a CPU (8-core Ryzen 7 2700X with 32 GB of RAM).

Figure 7: Runtime of T5-small performing inference on the ViGGO test set using different decoding methods and batch sizes. "No reranking" stands for simple beam search, while "Slot aligner" denotes beam search with slot aligner-based reranking. Model and data loading is excluded from the runtimes.

C Slot Aligner Details

For the purposes of the slot aligner, we classified slots into five general categories (*Boolean*, *numeric*, *scalar*, *categorical*, and *list*), covering the most common types of information MRs typically convey in data-to-text NLG. Each of these categories has its own method for extracting a slot mention from an utterance, generalized enough to be applicable across all slots in the category. This design allows for a straightforward extension of the slot aligner to a new domain, as it merely needs to be indicated which of the five categories each of the slots in the new domain belongs to. Optionally, it can be provided a simple dictionary of common alternatives for specific slot values, which tends to increase the slot aligner's performance.

Although a decreased matching accuracy – es-

pecially for rare slot realizations – is a trade-off for the scalable design, the slot aligner's typical application are generated model outputs, which get evaluated for semantic errors. There the slot aligner is not likely to encounter rare slot realizations frequently, if at all, due to the generalizing properties of neural NLG models. The rapid adaptability of the slot aligner to a new domain, on the other hand, is a very valuable feature.

C.1 Boolean Slots

Boolean slots take on binary values, such as "yes"/"no" or "true"/"false". Their realization in an utterance thus typically does not contain the actual value of the slot, but instead a mention of the slot name (e.g., "is a family-friendly restaurant" for FAMILYFRIENDLY[yes], or "not supported on Mac" for HAS_MAC_RELEASE[no]). Therefore, extracting a Boolean slot mention boils down to the following two steps: (1) finding a word or a phrase representing the slot, and (2) verifying whether the representation is associated with a negation or not.

The first step is straightforward, and only requires a list of possible realizations for each Boolean slot. This list rarely contains more than one element, which is the "stem" of the slot's name (e.g., "linux" for "HAS_LINUX_RELEASE"). It can thus be populated trivially for most of the new Boolean slots. And if a Boolean slot can have multiple equivalent realizations (such as "child friendly" or "where kids are welcome" for the slot FAMILYFRIENDLY), they are typically not numerous and can be listed manually. Having a list of stems (we refer to all the equivalent realizations of a slot collectively as "slot stems"), the utterance is scanned for the presence of each of them in it. If one is found, we go to the second step. A slot mention is decided to be negative if a negation cue is found to be modifying the slot stem in the utterance, and without a contrastive cue in between. It is decided to be positive if no negation cue is present within a certain distance of the stem, or there is a contrastive cue in between (see examples in Table 11).

C.2 Numeric Slots

Slots whose value is just a number (such as RELEASE_YEAR in ViGGO, or CHOICE in MultiWOZ) are in general not handled in any special way, and the value is simply matched directly in the utterance. However, there are certain numeric slot types that benefit from additional preprocessing: (1) those with a unit, and (2) years. When a numeric slot represents a year, the slot aligner generates the common abbreviated alternatives for the year (e.g., "'97" for the value "1997") that it tries to match in case the original value is not found in the utterance.

#1	There's *no* Linux release or multiplayer, **but** there is Mac support.
#2	**Though** it's *not available* on Linux, it does have a Mac release as well.
#3	It is available on PC and Mac **but** *not* Linux, and it can be found on Steam.

Table 11: Examples of contrastive phrases involving Boolean slots. Underlined are the stems of the Boolean slots for which the polarity is questioned. Note that in all 3 examples the mention is positive, despite the presence of contrast and negation distractors.

CUSTOMER RATING (E2E)	RATING (ViGGO)	Alternative expressions
low	poor	*bad, lacking, negative,...*
average	average	*decent, mediocre, okay,...*
-	good	*fun, positive, solid,...*
high	excellent	*amazing, fantastic, great,...*

Table 12: An example of value mapping between two similar scalar slots in the restaurant and video game domains.

C.3 Scalar Slots

Similarly to Boolean slot aligning, scalar slot aligning consist of two steps. The first one is the same, i.e., finding a word or a phrase representing the slot (which we refer to as "stem" in this case too, in order to maintain consistency). In the second step, however, the slot aligner looks for the slot's value, or its equivalent, occurring within a reasonable distance from the slot stem. The optional soft alignment mode skips the second step as long as a slot stem is matched in the first step.

We assume that scalar slots, even across different domains, will often have values that can be mapped to each other, as long as they are on the same or a similar scale (see Table 12). For each scalar slot, the slot aligner refers to a corresponding dictionary for possible alternative expressions of its value. With the above assumption, it is sufficient to have one dictionary per scale, or type of scale, which can be reused for similar scalar slots in different domains. The dictionaries can be quickly populated with synonyms of the values of a given scale (see the last column in the table), and thus

do not necessarily require manual additions every time the system is used with a new domain. Some alternative expressions might be suitable for scalar slots in some domains better than others, but that will not be an issue in most cases, since, being synonymous, they are not likely to cause conflicts, and the slot aligner will simply not encounter certain alternative expressions in certain domains.

C.4 Categorical Slots

Categorical slots can take on virtually any value. Nevertheless, for each such slot the values typically come from a limited, although possibly large, set of values. For instance, in the E2E dataset, the FOOD slot has 7 possible values, such as "Italian" and "Fast food", but technically it could take on hundreds of different values representing all of the cuisines of the world. Some values can be single-word, while others can have multiple words (e.g., "restaurant" and "coffee shop" as possible values for the EATTYPE slot). Due to this huge variety in possible values of categorical slots, the aligning methods need to remain very general.

Besides exact matching of the value in the utterance, the slot aligner can be instructed to perform the matching in three additional modes, besides exact, increasing its robustness while maintaining scalability. The four modes of aligning the slot with its mention work as follows:

- **Exact** - slot mention is identified only if it matches (case-insensitive) the slot value verbatim;
- **All words** - slot mention is identified if each of the value's tokens is found in the utterance, though they can be in an arbitrary order and they can be separated by other words;
- **Any word** - slot mention is identified by matching any of the value's tokens in the utterance;
- **First word** - slot mention is identified by matching just the value's first token in the utterance.

Note that for single-word values all four modes give the same result. The three non-exact modes offer different approaches to soft alignment for categorical slots. The choice may depend on the particular slot, and the mode can thus be specified for each slot separately, while by default the slot aligner operates in the exact-matching mode.

MR
inform(NAME [**BioShock**], DEVELOPER [**2K Boston**], GENRES [**action-adventure, role-playing, shooter**], HAS_MULTIPLAYER [**no**], PLATFORMS [**PlayStation, Xbox, PC**], HAS_LINUX_RELEASE [**no**], HAS_MAC_RELEASE [**yes**])
Reference utterance
Developed by **2K Boston**, **BioShock** is a **single-player shooter** game that will have you **role-playing** through a well constructed **action-adventure** narrative. It is **available for PlayStation**, **Xbox**, **Mac** and **PC**, but is **not available for Linux**.
Slot alignment
(**13:** DEVELOPER) (**25:** NAME) (**39:** HAS_MULTIPLAYER) (**53:** GENRES) (**174:** PLATFORMS) (**191:** HAS_MAC_RELEASE) (**228:** HAS_LINUX_RELEASE)

Table 13: An example from ViGGO that involves list slots. Notice how the individual value item mentions can be scattered across an entire sentence in a natural way. The bottom section indicates the slot mention positions determined by the slot aligner, given as the number of characters from the beginning of the utterance.

Similarly to Boolean and scalar slots, the slot aligner can search for alternative expressions of a value, if provided in the corresponding dictionary. The alternative matching is, however, more flexible here, as the alternatives in the dictionary can be multi-part, in which case the slot aligner tries to match all the parts (words/tokens/phrases) provided in the form of a list.

C.5 List Slots

A list slot is similar to a categorical slot, the only difference being that it can have multiple individual items in its value. Two instances of a list slot, namely GENRES and PLATFORMS, can be seen in the example from the ViGGO dataset in Table 13.

The aligning procedure for list slots thus heavily relies on that of categorical slots. In order to align a list slot with the corresponding utterance, the slot aligner first parses the individual items in the slot's value. It then iterates over all of them and performs the categorical slot alignment, as described in the previous section, with each individual item. Considering the items can be scattered over multiple sentences, the slot aligner considers the position of the leftmost mention of an item as the position of the corresponding list slot.

Author Index