ICNLSP 2022

Proceedings of the 5th International Conference on Natural Language and Speech Processing

16–17 December, 2022 (virtual)

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 978-1-959429-36-4

https://www.icnlsp.org

Introduction

Welcome to the 5th International Conference on Natural Language and Speech Processing (ICNLSP 2022), held online on December 16th, 17th 2022.

ICNLSP is the right choice to select as a forum for researchers, students, and also for industrials to exchange ideas and discuss research and trends in the field of Natural Language Processing, and also to publish their results in the field. As examples of companies present during the conference, we mention here, Mercedes Benz (Germany), and Vail Systems company (USA), and Elm (KSA) and many others.

The program committee accepted 37 papers (long and short ones) which is around 40% of the received submissions (from 31 countries). The accepted papers are of good quality thanks to the high-quality level of the reviews done by the program committee members. All papers have been presented orally, that is why the program was quite long. Various topics of NLP are discussed, as Semantics, language modelling, text classification, speech recognition, information extraction, natural language understanding, etc.

As it is mentioned in the program of the conference, there are three keynotes. The first one was presented by Prof. Eric Laporte from Gustave Eiffel University (France), who exposed his thoughts about hybrid natural language processing in the deep learning era. The second one, dealing with an interesting and challenging topic, was given by Dr. Ahmed Ali from Qatar Computing Research Institute (Qatar), entitled "Multilingual and Code-Switching Speech Recognition". The third talk was programmed to be presented by Prof. Jan Niehues, from Karlsruhe Institute of Technology (Germany), and entitled "Plug-and-Play Abilities for Neural Machine Translation". We will be happy to make all the talks and presentations available on the website of the conference.

We hope readers enjoy reading the content of the 5^{th} ICNLSP proceedings. We would like also to invite them to check the proceedings of the past versions of ICNLSP:

Mourad Abbas, Abed Alhakim Freihat, Proceedings of the Fourth International Conference on Natural Language and Speech Processing (ICNLSP 2021), 12-13 November 2021, Association for Computational Linguistics, https://aclanthology.org/2021.icnlsp-1

Mourad Abbas, Abed Alhakim Freihat, Proceedings of the 3rd International Conference on Natural Language and Speech Processing (ICNLSP 2019), 12-13 September 2019, Association for Computational Linguistics, https://aclanthology.org/volumes/W19-74/

Mourad Abbas, Proceedings of the 2nd International Conference on Natural Language and Speech Processing (ICNLSP 2018), 25-26 April 2018, IEEE, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8374402

Mourad Abbas, Ahmed Abdelali, Proceedings of the 1st International Conference on Natural Language and Speech Processing, Procedia Computer Science, 128, Elsevier. https://www.sciencedirect.com/journal/procedia-computer-science/vol/128

We would like to express our gratitude to the organizing and the program committees for making this event a success.

Mourad Abbas and Abed Alhakim Freihat

Organizers:

General Chair: Dr. Mourad Abbas

Chair: Dr. Abed Alhakim Freihat

Program Chair: Dr. Mourad Abbas

Publicity Chair: Dr. Muhammad Al-Qurishi

Program Committee:

Ahmed Abdelali, QCRI, Qatar.
Hend Al-Khalifa, King Saud University, Saudi Arabia.
Somaya Al-Maadeed, Qatar University, Qatar.
Muhammad Al-Qurishi, Elm, Saudi Arabia.
Yuan An, Drexel University, USA.
Fayssal Bouarourou, University of Strasbourg, France.
Markus Brückl, TU Berlin, Germany.
Hadda Cherroun, Amar Telidji University, Algeria.
Gérard Chollet, CNRS, France.
Dirk Van Compernolle, KU Leuven, Belgium.
Kareem Darwish, aiXplain, USA.
Najim Dehak, Johns Hopkins University, USA.
Abed Alhakim Freihat, University of Trento, Italy.
Munir Georges, Technische Hochschule Ingolstadt, Germany.
Fausto Giunchiglia, University of Trento, Italy.
Ahmed Guessoum, USTHB, Algeria.
Fouzi Harrag, Ferhat Abbas University, Algeria.
Valia Kordoni, Humboldt University, Germany.
Eric Laporte, Gustave Eiffel University, France.
Shang-Wen Li, Facebook AI., USA.
Mohamed Lichouri, USTHB, Algeria.
Mhamed Mataoui, EMP, Algeria.
Mohammed Mediani, University of Adrar, Algeria.
Fatiha Merazka, USTHB, Algeria.
Farid Meziane. University of Derby, UK.
Hamdy Mubarak, QCRI, Qatar.
Preslav Nakov, QCRI, Qatar.
Alexis Neme, UPEM, France.
Rasha Obeidat, Jordan university of science and technology, Jordan.
Axel Roebel, IRCAM, France.
Hassan Satori, Sidi Mohammed Ben Abbdallah University, Morocco.
Tim Schlippe, Silicon Surfer, Germany.
Nasredine Semmar, CEA, France.

Otakar Smrz, Džám-e Džam Language Institute, Czech Republic.
Rudolph Sock, University of Strasbourg, France.
R. V. Swaminathan, Amazon, USA.
Irina Temnikova, Big Data for Smart Society Institute, Bulgaria.
Jan Trmal, Johns Hopkins University, USA.
Rodrigo Wilkens, UCLouvain, Belgium.
Fayçal Ykhlef, CDTA, Algeria.
Wajdi Zaghouani, Hamad Bin Khalifa University, Qatar.
Hasna Zaouali, University of Strasbourg, France.

Organizing Committee:

Hadi Khalilia, University of Trento
Khaled Lounnas, USTHB, Algeria
Nandu C Nair, University of Trento

Invited Speakers:

Prof. Eric Laporte, Gustave Eiffel University, France
Dr. Ahmed Ali, Qatar Computing Research Institute, Qatar
Prof Jan Niehues, Karlsruhe Institute of Technology, Germany

Invited Talks

Hybrid natural language processing in the deep learning era

Prof. Eric Laporte, Gustave Eiffel University, France

In this talk, we examine critically the current wave of interest in pure deep learning for natural language processing. What can symbolic resources do for natural language processing? Among other examples, we take into account the languages with more restricted graphical delimitation than English. Then we discuss the foreseeable future of the synergy between machine learning and symbolic resources: are the goals of formalisation, precision, reliability, adaptability within reach for linguistic data?

Multilingual and Code-Switching Speech Recognition

Dr. Ahmed Ali, Qatar Computing Research Institute, Qatar

The prevalence of code-switching (CS) in spoken content has enforced automatic speech recognition (ASR) systems to handle mixed input. Yet, designing a CS-ASR has many challenges, mainly due to the data scarcity, grammatical structure complexity and mismatch along with unbalanced language usage distribution. Our CS will feature both intersentential (switching between-utterances) and intrasentential (within utterances). The evaluation of the designed system and the analysis of the phenomena will be driven based on real test cases, collected from real meetings and interviews.

We show our results on investigating novel techniques to build practical large vocabulary continuous speech recognition systems capable of dealing with both monolingual and code-switching spoken utterances. We study data augmentation and state of the art modelling techniques to address the lack of balanced transcribed CS data. Moreover, we investigate various challenges of evaluating code-switching ASR output. Finally, we highlight our effort in understanding where/why CS happens in speech analysis for system/human code-switching points.

Plug-and-Play Abilities for Neural Machine Translation

Prof Jan Niehues, Karlsruhe Institute of Technology, Germany

Advances in neural machine translation have led to impressive results and broad areas of application. Using multitask learning, these models have even abilities to process different input and generate a variety of output languages. However, this progress is often backed by millions of training examples. In order to cover the approximately 7000 languages in the words, it is essential to not only generalize to unseen examples, but also to unseen tasks. Therefore, we need to recombine the abilities of NMT systems to process and generate different languages in a plug-and-play fashion.

In this presentation, we will investigate two use cases: translating zero-shot directions in multilingual machine translation and end-to-end speech translation. First, we will dissect the challenges in the zero-shot condition. Motivated by the findings, we will present several methods to promote the possibility to combine the different abilities of an NMT system in order to perform unseen tasks. Finally, we will discuss the effect of the presented ideas on multi-lingual machine translation and speech translation.

Table of Contents

Error correction and extraction in request dialogs

Stefan Constantin and **Alex Waibel**
Karlsruhe Institute of Technology
Institute for Anthropomatics and Robotics
{stefan.constantin|waibel}@kit.edu

Abstract

We propose a dialog system utility component that gets the two last utterances of a user and can detect whether the last utterance is an error correction of the second last utterance. If yes, it corrects the second last utterance according to the error correction in the last utterance. In addition, the proposed component outputs the extracted pairs of reparandum and repair entity. This component offers two advantages, learning the concept of corrections to avoid collecting corrections for every new domain and extracting reparandum and repair pairs, which offers the possibility to learn out of it.

For the error correction one sequence labeling and two sequence to sequence approaches are presented. For the error correction detection these three error correction approaches can also be used and in addition, we present a sequence classification approach. One error correction detection and one error correction approach can be combined to a pipeline or the error correction approaches can be trained and used end-to-end to avoid two components. We modified the EPIC-KITCHENS-100 dataset to evaluate the approaches for correcting entity phrases in request dialogs. For error correction detection and correction, we got an accuracy of 97.54 % on synthetic validation data and an accuracy of 69.27 % on human-created real-world test data.

1 Introduction

Errors and ambiguities are difficult to avoid in a dialog. Corrections allow to recover from errors and to disambiguate ambiguities. For example, a household robot gets the request "Put the cleaned spoons into the cutlery drawer", but the robot does not know which one of the drawers is the cutlery drawer. It can choose one of the drawers and puts the spoons there. If its choice is wrong, the user must correct the robot, e. g. "No, into the drawer right of the sink". Alternatively, the robot can ask which one of the drawers is the cutlery drawer. The clarification response of the user, e. g. "It's the drawer right of the sink", is also a correction because the response disambiguates the ambiguity. Another type of correction occurs when the user changes their mind, e. g. "I changed my mind, the forks", or when the system misunderstands the user request (e. g. because of automatic speech recognition or natural language understanding errors).

All these correction types can be processed in the same manner and therefore we propose a component that gets a request and a correction and outputs a corrected request. To get this corrected request, the phrases in the correction phrase replace their corresponding phrases in the request. In this work, we restrict on entity phrases like "drawer right of the sink". To replace other phrases like verb phrases is out of scope for this work. The request "Put the cleaned spoons into the cutlery drawer" with its correction "No, into the drawer right of the sink" is converted to "Put the cleaned spoons into the drawer right of the sink". Such a component has two advantages compared to handling the corrections in the actual dialog component. First, it reduces the amount of required training data for the actual dialog component because corrections will not need to be learned if there is an open-domain correction component. Second, this kind of correction component can be extended so that it outputs the extracted pairs of reparandum and repair entity. In our example there is one pair: "cutlery drawer" and "drawer right of the sink". These entity pairs can be used, for example, for learning in a life-long learning component of a dialog system to reduce the need for correction in future dialogs, e. g. the robot can learn which one of the drawers is the cutlery drawer.

2 Related Work

Studies have been conducted in the area of interactive repair dialog. In (Suhm et al., 1996) a multimodal approach is used. The user can highlight

wrong phrases and respeak or spell the correct phrase, or choose from alternatives in the n-best list of the automatic speech recognition component, or use handwriting to write the correct phrase. These error strategies are improved in (Suhm and Waibel, 1997) by considering the context. In (Suhm et al., 1999, 2001) the previous approaches are evaluated in more detail in a dictation system with real users. Different human strategies for error correction are presented in (Gieselmann, 2006).

Sagawa et al. (2004) propose an error handling component based on correction grammars. These correction grammars have the advantage that they can be used domain-independently. However, they need a grammar based dialog system. An error correction detection module and strategies to handle the detected errors are proposed by Griol and Molina (2016). The corrected request must be handled by the Spoken Language Understanding component. That means, for every domain the Spoken Language Understanding component must be adapted to the possible corrections. Kraljevski and Hirschfeld (2017) propose a domain-independent correction detection by checking the speech for hyperarticulation. Other features than hyperarticulation are not used.

In (Béchet and Favre, 2013), a system is presented that detects errors in automatic speech recognition transcripts and asks the user for a correction.

There are also studies that research automatic error correction without user interaction. In (Xie et al., 2016) a character-based approach to correct language errors is used. They used a character-based approach to avoid out-of-vocabulary words because of orthographic errors. In (Weng et al., 2020), the authors used a multi-task setup to correct the automatic speech recognition outputs and do the natural language understanding.

The task of request correction presented in the introduction is related to the task of disfluency removal. In disfluency removal, there are the reparandum (which entity should be replaced), the interruption point (where the correction begins), the interregnum (which phrase is the signal phrase for the correction), and the repair phrase (the correct entity) (Shriberg, 1994).

In Figure 1, a disfluent utterance annotated with this terminology is depicted.

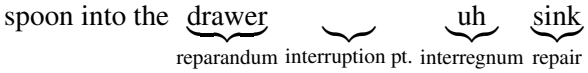


Figure 1: disfluent utterance annotated with repair terminology

A lot of work has been conducted for disfluency removal (Cho et al., 2014; Dong et al., 2019; Wang et al., 2016; Jamshid Lou et al., 2018). In all these works, it is assumed that it is enough to delete tokens of the disfluent utterance to get a fluent utterance. A disfluent utterance with the copy and delete labels is depicted in Figure 2.

spoons	into	the	drawer	uh	sink
C	C	C	D	D	C

Figure 2: disfluent utterance labeled with copy and delete labels

However, in the task of corrections, long-distance replacements can occur. That means, that between the reparandum and the repair are words that are important and must not be deleted. Such a long-distance replacement is depicted in Figure 3.

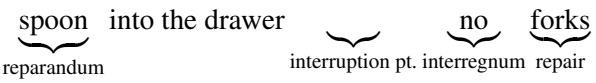


Figure 3: request and correction phrase annotated with repair terminology

3 Dataset

Our dataset is based on the annotations in natural language of the EPIC-KITCHENS-100 dataset (Damen et al., 2020, 2022). The EPIC-KITCHENS-100 dataset comprises 100 hours of recordings of actions in a kitchen environment. An example annotation of such an action is “put pizza slice into container” and the corresponding verb is “put-into” and the corresponding entities are “slice:pizza” and “container”. Annotations in this dataset have one verb and zero to six entities. The verb, the corresponding verb class, the entities and the corresponding entity classes are explicitly saved to every annotation. The order of the entities and the corresponding entity classes is the same as in the annotation. If the verb has a preposition, the verb is saved including its preposition. The words of the entities are represented in a hierarchy. The most general word of the hierarchy is left and the words

are more specialized the further to the right of the hierarchy. The words of each hierarchy are separated by a colon. There are 67 218 annotations in the training and 9669 annotations in the validation dataset of the EPIC-KITCHENS-100 dataset. There is no test dataset. Some annotations occur multiple times, because different recordings of the 100 hours recordings have the same annotation. By considering only the unique annotations, 15 968 annotations are in the training and 3835 annotations are in the validation dataset.

For our dataset, we used only the annotations that have one or two entities. We excluded the annotations with no entities because we need at least an entity that can be corrected. Annotations including more than two entities amount only to less than 1.15 % of all annotations and therefore we decided to exclude them because of dataset balancing reasons.

The verb classes of the EPIC-KITCHENS-100 datasets are imbalanced. To get a better balance in the validation dataset, we removed annotations of verb classes that occur very often from the validation dataset. We wanted a more balanced dataset to evaluate whether the model gets along with very different verb classes. We calculated the number of desired remaining annotations of a verb class, called r, by dividing the number of annotations, called a, by 100, but we determined a minimal number of remaining annotations of verb classes: 2 for one entity annotations ($r = max(2, a/100)$) and 4 for two entity annotations ($r = max(4, a/100)$). In some cases, there are less than the desired remaining annotations of a verb class in the EPIC-KITCHENS-100 dataset. We then used the possible number. We chose the values for minimal examples to get a nearly balanced dataset: 142 annotations with one entity and 122 annotations with two entities. To get the annotations of a verb class, we chose the verbs occurring in a verb class equally distributed. In total, we have 264 annotations in the reduced validation dataset. The number of unique annotations in respect to the verb class before and after the reduction are depicted in Figure 4.

In the EPIC-KITCHENS-100 dataset, the training and validation datasets are similar: all 78 verb classes of the validation dataset occur in the training dataset and 346 of the 372 first level word of the entity hierarchies of the validation dataset occur in the training dataset. Because of this, we decided to reduce the training dataset to have more difference

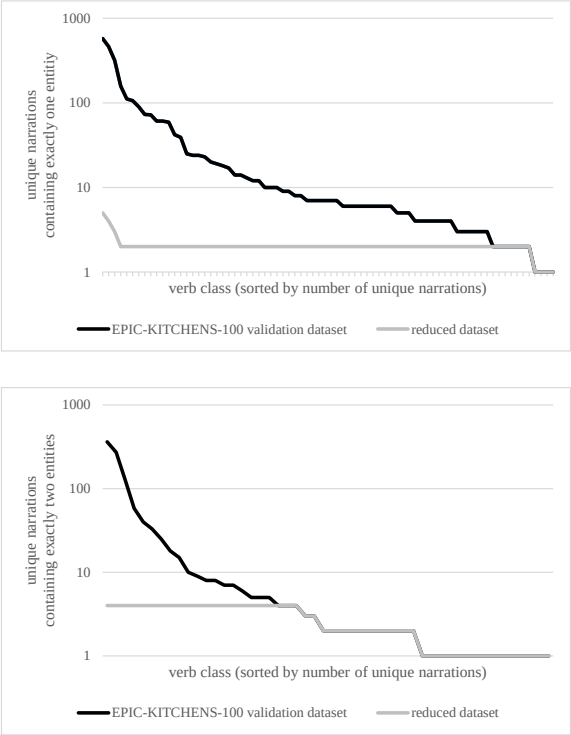


Figure 4: unique annotations in respect to the verb class before and after the reduction of the EPIC-KITCHENS-100 validation dataset

between them. We removed the verb classes of the 49 less frequent occurring verb classes (in total 98 verb classes are in the training dataset) from the training dataset and removed all entities from the training dataset when its first part was also in the validation dataset. That means, if bowl:washing:up was in the validation dataset, an annotation with bowl:salad in the training dataset was removed. After the reduction 4822 annotations were left in the training dataset.

To use these annotations for training and evaluating the error correction detection and correction component, we had to add corrections to the annotations. For the training and validation dataset, we generated the corrections synthetically. There are three options for the entitiy replacement: the first entity should be replaced, the second entity should be replaced, or both entities should be replaced. We drew uniformly distributed which of these three options should be applied. If both entities should be corrected, we drew uniformly distributed in which order they should be corrected. For the training and validation dataset, we had 8 and 6, respectively, templates to introduce the correction phrase, followed by the corrected entities. An entity could be

replaced by an entity that occurs in an annotation of the same verb class in the same position. An example for one corrected entity is “Be so kind and pick the oregano” for the request and “it’s the chilli” for the correction and an example for two corrected entities is “Could you put the tin in the Cupboard?” for the request and “no the olives in the Fridge” for the correction.

For the test dataset, we had nine human data collectors who could freely write the corrections, they only knew what entities should be replaced with what other entities (but were allowed to use synonyms for the other entities) and whether the correction should be a correction to a wrong action of the robot, a clarification, or a correction because the user changed their mind (equally distributed).

We added 19 and 14 templates before the narration to increase the variety of the natural language of the training and validation dataset, respectively. In the EPIC-KITCHENS-100 dataset, the articles of the entities are missing, therefore we added a “the” before the entities. For the test dataset, we used the narrations of our validation dataset and let the same nine annotators that created the corrections for our test dataset paraphrase them.

The test dataset is more challenging than the validation dataset because it differs even more from the training dataset. The nine data collectors were told to use a large variety of natural language.

We used the 4822 annotations of the reduced training dataset to generate with the different data augmentations 52 357 request and correction pairs for the error correction training dataset. The error correction validation dataset has 264 request and correction pairs and the error correction test dataset has 960 request and correction pairs.

To train and evaluate the error correction detection, we need examples where the last utterance is no correction. To achieve this, the second last and the last utterance are made of all the requests of the error correction data. The requests were shuffled for the last utterance. This approach doubled the number of examples to the correction examples, that means, we have 104 714 pairs in the error correction detection and error correction training dataset, 528 pairs in the error correction detection and error correction validation dataset, and 1920 pairs in the error correction detection and error correction test dataset.

The target for the error correction datasets is the corrected request and the reparandum repair pairs and the target for the error correction detection and error correction dataset depends whether the source has a request and correction pair or a request and request pair. In the first case, there is an error correction and the target is the same as in the error correction datasets, in the second case, the target is to copy both requests. There is a further dataset, the error correction detection dataset. The sources are the same as in the error correction detection and error correction dataset but the target is the binary value whether there is a correction or not.

We created the described datasets in different forms for the different approaches. For the sequence labeling approach, we labeled the source tokens with different labels, see Figure 5 and Section 4 for an explanation of the labels.

For the sequence to sequence approach with generative token generation, we created source and target pairs, see Figure 6. For the sequence to sequence approach with generation by copying source tokens, we added the order of copy operations. Additionally, the separator tokens that are needed in the target will be inserted to the source, see Figure 7.

```
would C
it C
be C
possible C
to C
wash C
the C
table R1
? C
| D
no D
the D
wok S1
instead D
of D
the D
table D
. D
```

Figure 5: sequence labeling data example

```
source file: Would it be possible to wash the table
? | no the Wok instead of the table .
target file: Would it be possible to wash the Wok ?
| table -> Wok
```

Figure 6: sequence to sequence with fixed vocabulary data example

source file: Would it be possible to wash the table ? | no the Wok instead of the table . - ->
target file: Would it be possible to wash the Wok ? | table -> Wok
copy target file (considering the T5 prefix and the T5 tokenization): 3 4 5 6 7 8 9 16 17 11 12 13 10 26 27 16 17 28

Figure 7: sequence to sequence with copy source token approach data example

4 Models

For the error correction and extraction, we developed three different approaches. The first approach is a sequence labeling approach, the second approach is a sequence to sequence approach where the output tokens are sampled from a fixed vocabulary, and the third approach is a sequence to sequence approach where output tokens are copied from the source tokens.

For the sequence labeling approach, every word is labeled with one of the following labels: C (copy), D (delete), R1 (entity 1 potentially to be replaced), R2 (entity 2 potentially to be replaced), S1 (entity to replace entity 1), or S2 (entity to replace entity 2). For the correction target, the S1 and S2 labeled entities are used to replace the R1 and R2 labeled entities, respectively. For the extraction target, the output is the pairs R1 and S1 as well as R2 and S2 if there is a replacement available for the first or second entity, respectively. In Figure 8, an example request and correction pair is labeled and both targets are given.

For the sequence labeling, we propose fine-tuning the cased BERT large model (24 Transformer encoder blocks, hidden size of 1024, 16 self-attention heads, and 340 million parameters) (Devlin et al., 2019).

For the sequence to sequence approach where the output tokens are sampled from a fixed vocabulary, we propose fine-tuning a T5 large model (Raffel et al., 2020). The T5 model is a pre-trained Transformer network (Vaswani et al., 2017) and the T5 large model has the following properties: 24 Transformer encoder blocks, 24 Transformer decoder blocks, hidden size of 1024 (in- and output) and 4096 (inner-layer), 16 self-attention heads, 737 million parameters.

The probability distribution over the fixed vocabulary V can be calculated in the following way:

$$P_{\text{generate}}(V) = softmax(dec^T \cdot W_{\text{generate}})$$

where dec is the output of the Transformer decoder and $W_{\text{generate}} \in \mathbb{R}^{\text{hidden size decoder} \times \text{vocabulary size}}$ is a learnable matrix.

We call this T5 model T5 generate.

In the corrected request there are only tokens of the input sequence. To utilize this property, we developed a pointer network model (Vinyals et al., 2015) with the T5 large model that calculates which input token has the highest probability to be copied to the output sequence. This is our third approach. The probability distribution over the input sequence tokens V' can be calculated in the following way:

$$P_{\text{copy}}(V') = softmax(dec^T \cdot enc^T)$$

where dec is the output of the Transformer decoder and $enc \in \mathbb{R}^{\text{source input length} \times \text{embedding size}}$.

To utilize the knowledge of the pre-trained model, we feed the source input token with the highest probability into the encoder instead of the position of the source input token. That means, that in the generation stage the copy mechanism is only used, otherwise it is like a normal T5 model. To be able to output the separators, we add this to the source, so that they can also be copied. We call this modificated T5 model T5 copy.

To decide whether an utterance is a correction for the previous request command, the described three approaches can also be used. If all output labels of the sequence labeling approach are C, no error correction is detected, otherwise there is an error correction. The sequence to sequence approaches detect an error correction if the source and the target without the separators are not equal, otherwise there is no error correction. In the T5 copy approach, the source for the comparison is the original source and not the source with the inserted separators.

In addition to these three approaches, a sequence classification can also be used for the error correction detection. For the sequence classification, we propose to fine-tune the cased BERT large model (24 Transformer encoder blocks, hidden size of 1024, 16 self-attention heads, and 340 million parameters) (Devlin et al., 2019).

5 Implementation

We used the HuggingFace (Wolf et al., 2020) Pytorch (Paszke et al., 2019) BERT and T5 models for our implementations of the models described in Section 4 and published our implementations and

request ‖ correction	put	the	milk	into	the	shelf	no	the	soja	milk	into	the	left	shelf
labels	C	R1	R1	R2	R2	R2	D	S1	S1	S1	S2	S2	S2	S2
corrected request	put the soja milk into the left shelf													
pairs of reparandum and repair entity	milk → soja milk - into the shelf → into the left shelf													

Figure 8: error correction example

our models [1].

6 Evaluation

In this section, we will first evaluate the different error correction detection component approaches described in Section 4. After that, the error correction component approaches described in Section 4 are evaluated. Third, we will compare whether it is better to separate the error correction detection and error correction in separate components and use a pipeline approach or whether an end-to-end approach is better. For all evaluations, we used the datasets described in Section 3.

We fine-tuned the sequence classification and labeling approaches one epoch with the following hyperparameters: AdamW optimizer (Loshchilov and Hutter, 2019) with learning rate of $2 \cdot 10^{-5}$, batch size of 32 and maximum input length of 128.

The T5 generate and T5 copy models were fine-tuned one epoch with the following hyperparameters: Adam optimizer (Kingma and Ba, 2015) with learning rate of $2.5 \cdot 10^{-4}$, batch size of 24 and a maximum input length of 12; in the embedding layer, the first two encoder blocks were frozen.

The results of the error correction detection components are depicted in Table 1. Accuracy means how many examples were classified correctly, precision is how many of the positive classified examples are really positive, recall how many of the positive examples are found by the component and the F_1-score is the harmonic mean of the precision and recall. We calculated the precision, recall and F_1-score for the case that detecting corrections were the positive examples and for the case that detecting no corrections were the positive examples to get better insights in the quality of the differently trained models. The sequence classification approach was trained with the error correction detection dataset and the other approaches were trained with the error correction detection and error correction dataset. The best approach is the sequence labeling approach (if all words have the copy label C, it is no error correction, otherwise it is an error correction). It has an accuracy of 100 % for the validation and 88.49 % for the test dataset. The recall for detecting no corrections is 99.90 % and the precision 81.34 % (F_1-score 89.67 %) in the test dataset. That means, if there is no correction, the component detects it in most of the cases and make no unnecessary correction. This is a good property, because it is better not detecting a correction than correcting something which is already correct. The error correction detection and error correction component should improve the overall system and not make it worse. Nevertheless, the results for detecting corrections with a recall of 77.08 % and a precision of 99.87 % (F_1-score 87.01 %) in the test dataset are good. In some cases where the component fails, it is really difficult to detect the correction like in "Kindly turn off the heat on the oven | Please turn off the water tap on the oven". The classification approach has similar results to the sequence labeling approach: 100 % accuracy for the validation dataset and 87.86 % for the test dataset. This approach also prefers detecting no corrections over corrections. The T5 generate approach is worse. It has an accuracy of 98.67 % on the validation dataset and an accuracy of 84.01 % on the test dataset. The worst results are from the T5 copy approach (71.78 % and 77.45 % validation and test dataset accuracy, respectively).

The results of the error correction components are depicted in Table 2. We evaluated the error correction with the metric accuracy. The correction is correct if the predicted correction and the reference correction are the same. The extraction of the reparandum and repair pairs is correct if the predicted pairs are equal to the reference pairs. The order and entities that map to themselves are ignored. Both are correct if the correction as well as the extraction are correct. For this evaluation the error correction datasets are used. On the validation dataset, the sequence labeling approach that

[1] https://github.com/msc42/seq2seq-transformer https://github.com/msc42/seq-labeling-and-classification

			detecting corrections			detecting no corrections		
dataset	model	accuracy	precision	recall	F_1-score	precision	recall	F_1-score
valid.	classification	100 %	100 %	100 %	100 %	100 %	100 %	100 %
valid.	seq. labeling	100 %	100 %	100 %	100 %	100 %	100 %	100 %
valid.	T5 generate	98.67 %	98.13 %	99.24 %	98.68 %	99.23 %	98.11 %	98.67 %
valid.	T5 copy	71.78 %	63.92 %	100 %	77.99 %	100 %	43.56 %	60.69 %
test	classification	87.86 %	99.86 %	75.83 %	86.20 %	80.52 %	99.90 %	89.17 %
test	seq. labeling	88.49 %	99.87 %	77.08 %	87.01 %	81.34 %	99.90 %	89.67 %
test	T5 generate	84.01 %	96.58 %	70.52 %	81.52 %	76.78 %	97.50 %	85.91 %
test	T5 copy	77.45 %	73.63 %	85.52 %	79.13 %	82.73 %	69.38 %	75.47 %

Table 1: evaluation results of the error correction detection, all models except the classification were trained on the error correction detection and error correction dataset and the classification was trained on the error correction detection dataset

	validation dataset			test dataset		
model	correction	extraction	both	correction	extraction	both
seq. labeling	96.21 %	94.70 %	94.70 %	40.10 %	48.75 %	39.06 %
E2E seq. labeling	96.59 %	95.08 %	95.08 %	39.27 %	43.54 %	38.65 %
T5 generate	92.80 %	95.83 %	91.29 %	73.65 %	77.81 %	71.98 %
E2E T5 generate	96.21 %	95.08 %	94.70 %	37.40 %	38.75 %	36.25 %
T5 copy	50.38 %	87.12 %	50.00 %	50.52 %	62.19 %	47.92 %
E2E T5 copy	70.83 %	92.42 %	68.94 %	27.50 %	35.00 %	25.31 %

Table 2: evaluation results of the error correction (metric accuracy), the end-to-end (E2E) models were trained on the error correction detection and error correction dataset and the other models were trained on the error correction dataset

model(s)	validation dataset			test dataset		
	correction	extraction	both	correction	extraction	both
detection and seq. labeling	96.21 %	94.70 %	94.70 %	34.27 %	36.88 %	33.54 %
detection and E2E seq. labeling	96.59 %	95.08 %	95.08 %	39.27 %	43.54 %	38.65 %
E2E seq. labeling	98.30 %	97.54 %	97.54 %	69.58 %	71.77 %	69.27 %
classification and T5 generate	92.80 %	95.83 %	91.29 %	56.98 %	60.21 %	56.04 %
classification and E2E T5 generate	96.21 %	95.08 %	94.70 %	36.67 %	38.23 %	35.73 %
E2E T5 generate	97.54 %	97.16 %	96.40 %	68.07 %	68.49 %	66.88 %
classification and T5 copy	50.38 %	87.12 %	50.00 %	36.98 %	46.88 %	35.21 %
classification and E2E T5 copy	70.83 %	92.42 %	68.94 %	26.67 %	34.38 %	24.79 %
E2E T5 copy	69.70 %	78.03 %	56.25 %	55.00 %	58.91 %	47.40 %

Table 3: evaluation results of the error correction detection and error correction (metric accuracy), the end-to-end (E2E) models were trained on the error correction detection and error correction dataset and the other models were trained on the error correction dataset, “and” means that the error correction detection was done by the best error correction detection model (sequence labeling) and the error correction detection by the model mentioned after the “and” if a correction was detected

was trained on the error correction detection and error correction datasets has the best overall accuracy (95.08 %). The accuracy for the correction is 96.59 % and for the extraction 95.08 %. On the test dataset, the T5 generate approach that is trained on the error correction dataset has the best accuracy (71.98 %). In general, all approaches trained on the error correction detection and error correction dataset have a higher accuracy on the validation dataset and all approaches trained on the error correction dataset have a higher accuracy on the test dataset. The T5 copy extraction could be optimized by bookkeeping the order of copy operations, stopping after finishing the correction and use the bookkeeping to reconstruct the reparandum and repair pairs. We relinquished this optimization because the correction results were much worse and we did not see any sense in further optimizations that will only lead to minimal improvements.

The results of the error correction detection and error correction components are depicted in Table 3. We used the same metric accuracy as in the error correction evaluation. For the error correction detection in the pipeline approach, we used the best error correction detection model evaluated in this section. It is the sequence labeling approach where no correction is in the example if all labels are C. After the error correction detection, the error correction will occur. We evaluated all three approaches described in Section 4 in their version trained on the error correction detection and error correction dataset and their version trained on the error correction dataset. In the end-to-end setting, a component executes the error correction detection and the error correction in one run. The results are that the end-to-end approaches are better than the pipeline approaches except for the end-to-end T5 copy approach for the validation dataset because of its bad error correction detection results on the validation dataset. The best approach is the end-to-end sequence labeling approach with an accuracy of 97.54 % on the validation and 69.27 % accuracy on the test dataset. This approach also has the best results in the error correction detection and error correction of the end-to-end approaches and therefore it is clear that it is the best approach for the combined error correction detection and error correction. However, the end-to-end T5 generator approach is not much worse with 96.40 % and 66.88 % validation and test accuracy, respectively.

The evaluation results show that the test dataset is more challenging than the validation dataset. The nine data collectors were able to introduce even more variety of natural language than the validation dataset has.

7 Conclusions and Further Work

The proposed error correction detection and error correction component shows high potential. For the validation dataset, we got very good results: in 97.54 % of the cases, we could detect whether there is a correction or not and if there is a correction, it outputs a correct corrected request and could extract correctly the reparandum and repair pairs. The results for the human-generated real-world data with 69.27 % shows that the proposed component is learning the concept of corrections and can be developed to be used as an upstream component to avoid the need for collecting data for request corrections for every new domain. In addition, the extraction of the pairs of reparandum and repair entity can be used for learning in a life-long learning component of a dialog system to reduce the need for correction in future dialogs.

In future work, the training dataset could be extended to a bigger variety of natural language which will enable the model to learn the concept of corrections better and to get better results on human-generated real-world data. The mentioned life-long learning component could also be part of future work and the classification of correction types could improve the performance of such a life-long learning component. To improve the accuracy, architectures that have a better NER performance than our used BERT model, like the architecture proposed by (Baevski et al., 2019), could be used. A further future research goal is to be able to correct all phrases and not only entity phrases.

Acknowledgements

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the project OML (01IS18040A).

References

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. 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)*.

Frédéric Béchet and Benoit Favre. 2013. Asr error segment localization for spoken recovery strategy. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6837–6841.

Eunah Cho, Jan Niehues, and Alex Waibel. 2014. Machine translation of multi-party meetings: Segmentation and disfluency removal strategies. In *Proceedings of the 11th International Workshop on Spoken Language Translation (IWSLT)*.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. 2020. The EPIC-KITCHENS dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. 2022. Rescaling egocentric vision: Collection, pipeline and challenges for EPIC-KITCHENS-100. *International Journal of Computer Vision (IJCV)*, 130(1):33–55.

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 (NACL)*.

Qianqian Dong, Feng Wanga, Zhen Yang, Wei Chenand Shuang Xu, and Bo Xu. 2019. Adapting translation models for transcript disfluency detection. In *Proceedings of the Thirty-Third Conference on Artificial Intelligence (AAAI)*.

Petra Gieselmann. 2006. Comparing error-handling strategies in human-human and human-robot dialogues. In *Proceedings of the 8th Conference on Natural Language Processing (Konferenz zur Verarbeitung natürlicher Sprache, KONVENS)*.

David Griol and José Manuel Molina. 2016. A framework for improving error detection and correction in spoken dialog systems. *Soft Computing*, 20:4229–4241.

Paria Jamshid Lou, Peter Anderson, and Mark Johnson. 2018. Disfluency detection using auto-correlational neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR), Conference Track Proceedings*.

Ivan Kraljevski and Diane Hirschfeld. 2017. Hyperarticulation of corrections in multilingual dialogue systems. In *Proceedings of the 18th Annual Meeting of the International Speech Communication Association (Interspeech)*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations (ICLR)*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

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.

Hirohiko Sagawa, Teruko Mitamura, and Eric Nyberg. 2004. Correction grammars for error handling in a speech dialog system. In *Proceedings of HLT-NAACL 2004: Short Papers*.

Elizabeth Ellen Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis, University of California.

Bernhard Suhm, Brad A. Myers, and Alex Waibel. 1996. Interactive recovery from speech recognition errors in speech user interfaces. In *The 4th International Conference on Spoken Language Processing (ICSLP)*.

Bernhard Suhm, Brad A. Myers, and Alex Waibel. 1999. Model-based and empirical evaluation of multimodal interactive error correction. In *Proceeding of the CHI '99 Conference on Human Factors in Computing Systems: The CHI is the Limit*.

Bernhard Suhm, Brad A. Myers, and Alex Waibel. 2001. Multimodal error correction for speech user interfaces. *ACM Transactions on Computer-Human Interaction*, 8(1):60–98.

Bernhard Suhm and Alex Waibel. 1997. Exploiting repair context in interactive error recovery. In *Fifth European Conference on Speech Communication and Technology (EUROSPEECH)*.

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 30*, pages 5998–6008. Curran Associates, Inc.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Shaolei Wang, Wanxiang Che, and Ting Liu. 2016. A neural attention model for disfluency detection. In *Proceedings of the 26th International Conference on Computational Linguistic (COLING)*.

Yue Weng, Sai Sumanth Miryala, Chandra Khatri, Runze Wang, Huaixiu Zheng, Piero Molino, Mahdi Namazifar, Alexandros Papangelis, Hugh Williams, Franziska Bell, and Gökhan Tür. 2020. Joint contextual modeling for ASR correction and language understanding. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

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 (EMNLP): System Demonstrations*.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. 2016. Neural language correction with character-based attention. *CoRR*, abs/1603.09727.

Efficient Task-Oriented Dialogue Systems with Response Selection as an Auxiliary Task

Radostin Cholakov
High School of Mathematics
Plovdiv, Bulgaria
r.cholakov@obecto.com

Todor Kolev
Obecto Ltd.
Sofia, Bulgaria
tkolev@obecto.com

Abstract

The adoption of pre-trained language models in task-oriented dialogue systems has resulted in significant enhancements of their text generation abilities. However, these architectures are slow to use because of the large number of trainable parameters and can sometimes fail to generate diverse responses. To address these limitations, we propose two models with auxiliary tasks for response selection - (1) distinguishing distractors from ground truth responses and (2) distinguishing synthetic responses from ground truth labels. They achieve state-of-the-art results on the MultiWOZ 2.1 dataset with combined scores of 107.5 and 108.3 and outperform a baseline with three times more parameters. We publish reproducible code and checkpoints and discuss the effects of applying auxiliary tasks to T5-based architectures.

1 Introduction

Task-oriented dialogue (TOD) systems are developed to lead conversations with users and assist them with the completion of various tasks. Unlike traditional solutions which rely on natural language understanding, state tracking, language generation, and other modules, end-to-end systems utilize a single network for all required functionality (Young et al., 2013). The recent research in the field has concentrated on leveraging language models pre-trained on general-domain corpora (Devlin et al., 2018; Radford et al., 2019; Raffel et al., 2020) to produce more robust architectures fine-tuned specifically for TOD generation. This has bridged the gap between production-ready modularized pipelines and single-network models in terms of accuracy and human-sounding results. However, such architectures are big and computationally expensive; they are also prone to overfitting on the final task and "forgetting" useful capabilities from the pre-training phase (Greco et al., 2019; Kulhánek et al., 2021). Multiple studies (Section 2) have demonstrated that learning related auxiliary tasks can improve the generation performance of a model while making it less affected by the overfitting issue.

In this paper, we study the effects of learning auxiliary response selection tasks together with an architecture based on the T5 (Raffel et al., 2020) text-to-text transformer. We use MTTOD (Lee, 2021), trained on the MultiWOZ 2.1 (Eric et al., 2019) dataset, as a baseline and evaluate two main approaches for response selection:

- Binary classifier to distinguish between encodings of ground truth responses and encodings of distractor sentences sampled from the dataset.
- Binary classifier to tell apart ground truth responses from decoder-generated sequences.

Reproducible code and model checkpoints are available at https://github.com/radi-cho/RSTOD.

2 Related Work

TOD sequence-to-sequence models usually generate a belief state based on the dialogue history and then use the belief state (in addition to the previous context) to generate a response (Lei et al., 2018).

Pre-trained Language Models such as BERT (Devlin et al., 2018), GPT-2 (Radford et al., 2019), and T5 (Raffel et al., 2020) significantly enhance dialogue systems when they are fine-tuned for sequence tasks. The first study to validate this on GPT-2 is (Budzianowski and Vulić, 2019). SOLOIST (Peng et al., 2020), UBAR (Yang et al., 2021), and SimpleTOD (Hosseini-Asl et al., 2020) further develop the end-to-end setting of the problem by considering database results and generated responses during training. MinTL (Lin et al., 2020) provides a minimalistic **transfer learning** dialogue system with multiple backbones. TOD-BERT (Wu

et al., 2020) utilizes a contrastive objective function to mimic a response selection task. (Yang et al., 2022) augments data by ignoring nonessential tokens and also adversarially filters "easy" samples to enhance model robustness.

Auxiliary Learning - training additional tasks which improve the performance of the primary text generation task - is increasingly applied in TOD systems. AuGPT (Kulhánek et al., 2021) demonstrates that response selection tasks are helpful on top of GPT-2. MTTOD (Lee, 2021) has a span selection auxiliary task. GALAXY (He et al., 2022) (with UniLM (Dong et al., 2019) as a backbone) optimizes four objectives, one of which is a selection between ground truth responses and randomly sampled responses. PPTOD (Su et al., 2022) is also trained for multiple tasks in a plug-and-play fashion (Dathathri et al., 2019).

3 Method

3.1 Dialogue System Baseline

As a baseline, we use the end-to-end system setting introduced in (Lee, 2021) (Figure 1) with T5 encoder-decoder backbone. The encoder input consists of a dialogue history concatenated with a user utterance. A *belief state* decoder generates a sequence of a domain name, slot names, and slot values. There is an option for querying a domain-specific database based on the belief state to generate a *DB state* which is then used to condition a final *response* decoder. The response output contains a *system action* state - a sequence of a domain name, action types and slots - and a *system response*. Since the decoder works autoregressively[1], response generation is automatically conditioned on the system action.

As shown in figure 1, MTTOD utilizes a classifier as an auxiliary task for span matching, inspired by recent dialogue state tracking approaches. Labels for this task are the extractive informable slots defined in (Gao et al., 2020).

The loss to be jointly minimized is

$$\mathcal{L} = \alpha\,\mathcal{L}_{belief} + \beta\,\mathcal{L}_{resp} + \gamma\,\mathcal{L}_{span} \qquad (1)$$

where $\mathcal{L}_{belief}$ and $\mathcal{L}_{resp}$ are negative log-likelihood language modeling losses for the two decoders and $\mathcal{L}_{span}$ is a cross-entropy loss for the span task. For compatibility with (Lee, 2021) the coefficients α, β and γ are set to 1.0, 1.0 and 0.5 respectively. Refer to section 5 for baseline benchmarks.

[1] An autoregressive decoder uses information from previous time steps to generate the value at the current time step.

3.2 Response Selection as an Auxiliary Task

Our study aims to evaluate the effects of using response selection as an additional auxiliary task for the presented T5-based dialogue system. We propose two variants for such a task (Figure 2) and modify the full objective to

$$\mathcal{L} = \alpha\,\mathcal{L}_{belief} + \beta\,\mathcal{L}_{resp} + \gamma\,\mathcal{L}_{span} + \delta\,\mathcal{L}_{select} \qquad (2)$$

In our experiments δ is also set to 0.5.

3.2.1 Distinguishing distractor encodings

The first proposal for a response selection task in our system is a binary classifier head - a linear layer or a simple multilayer perceptron - distinguishing randomly sampled *distractor* responses from ground truth responses. During training, the dialogue context C_t at time step t (consisting of history H_t and user utterance U_t) is concatenated with both the ground truth labels T_t - forming a sequence (C_t, T_t) - and a distractor response D_t sampled from the dataset - forming a sequence (C_t, D_t). Encodings for both sequences are generated by the already implemented T5 encoder and are then fed to the response selection head. The class label is 0 for (C_t, D_t) and 1 for (C_t, T_t). The binary cross entropy loss to be minimized is defined as

$$\begin{aligned}\mathcal{L}_{select} = &-\log p\,(l = 1 \mid C_t, T_t) \\ &-\log p\,(l = 0 \mid C_t, D_t)\end{aligned} \qquad (3)$$

$$\begin{aligned}&p\,(l = 1 \mid C_t, T_t) = \\ &\text{sigmoid}\,(\phi_a(\phi_E(C_t, T_t))) \in \mathbb{R}^1 \\ &p\,(l = 0 \mid C_t, D_t) = \\ &1 - \text{sigmoid}\,(\phi_a(\phi_E(C_t, D_t))) \in \mathbb{R}^1\end{aligned} \qquad (4)$$

where ϕ_E denotes the encoder and ϕ_a - the final classifier.

Optimizing the auxiliary response selection task affects the gradients of the encoder parameters. We empirically prove that this is beneficial for the overall score improvements on multiple metrics.

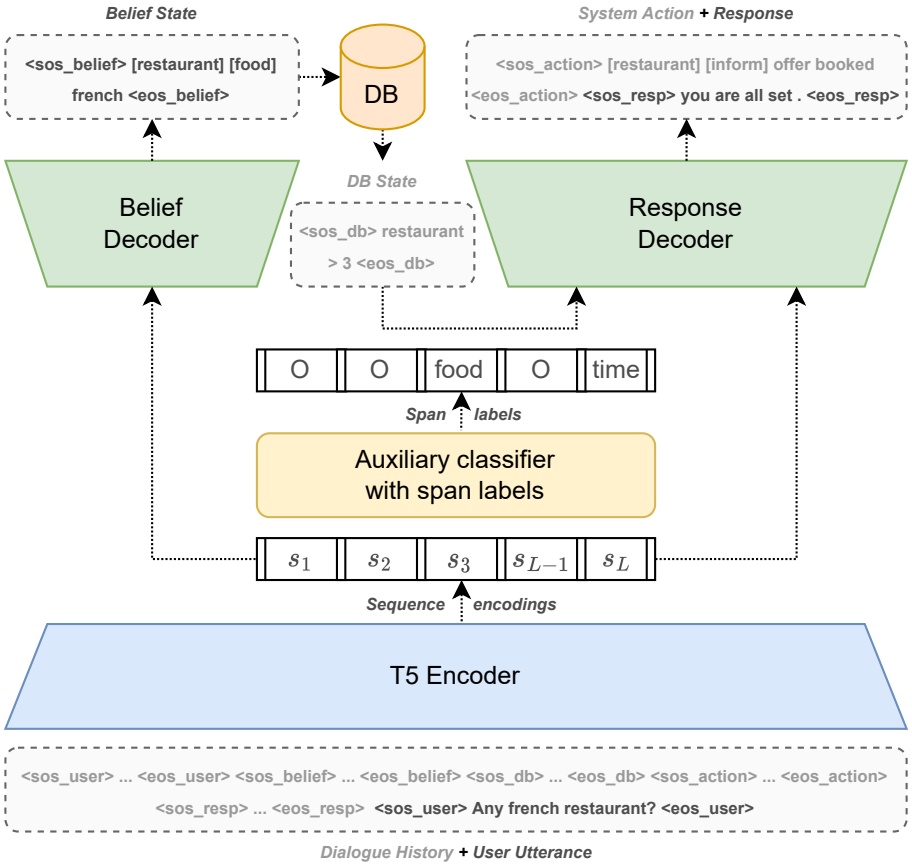


Figure 1: Dialogue generation architecture.

3.2.2 Distinguishing generated sequences

We also propose another independent auxiliary task for response selection inspired by Generative Adversarial Networks (Goodfellow et al., 2014). Its goal is to distinguish between responses from the transformer R_t and ground truth sequences T_t.

The baseline response decoder generates a sequence of token logits $\pi_1, \pi_2, ..., \pi_k$, where π_i is a vector of unnormalized class outputs over a vocabulary with size v. To obtain token ids we usually apply

$$\arg\max_j [\log \pi_{ij}], \quad j \in [1, v-1] \tag{5}$$

for every π_i. However, such a step is not differentiable, and when subsequent layers are optimized, transformer gradients won't be affected, making the auxiliary task useless. One way to overcome the limitation is to re-encode the sequences as previously described in 3.2.1 and thus backpropagate knowledge to the T5 encoder. Instead, we propose a classifier that works with differentiably sampled token representations and backpropagates knowledge to the whole architecture during training.

We sample vocabulary class probabilities y_{i1}, y_{i2}, ..., y_{iv} for every token representation π_i from a Gumbel-Softmax distribution (Jang et al., 2016; Maddison et al., 2016; Gumbel, 1954):

$$y_{ij} = \frac{\exp((\log(\pi_{ij}) + g_j)/\tau)}{\sum_{k=1}^{v} \exp((\log(\pi_{ik}) + g_k)/\tau)} \tag{6}$$

where τ is a temperature, treated as a hyperparameter, and g_j is a noise sample from a Gumbel$(0, 1)$ distribution which can be computed by drawing a $u \sim$ Uniform$(0, 1)$ and applying

$$g = -\log(-\log(u)) \tag{7}$$

For consistency with ground truth response sequences which are represented with v-dimensional one-hot vectors $\hat{y}_i$, we programmatically[2] convert the probabilities y_i to one-hot vectors but compute gradients with their continuous values.

[2]Refer to the *hard* flag in http://pytorch.org/docs/stable/generated/torch.nn.functional.gumbel_softmax

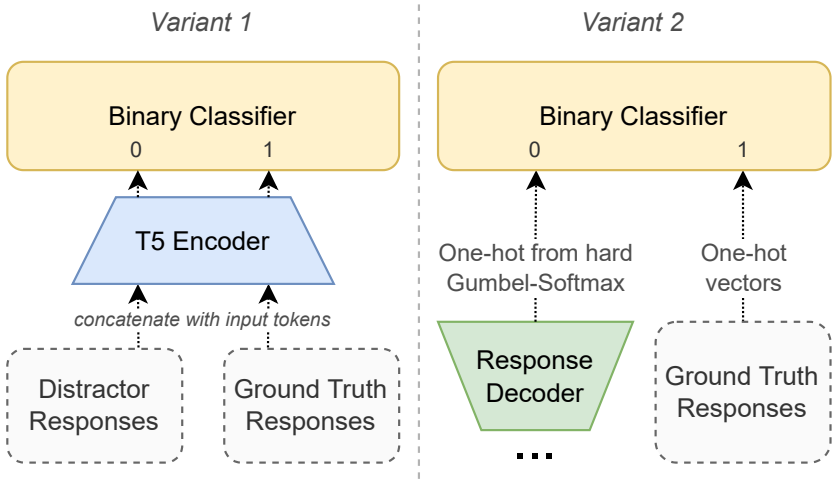


Figure 2: Binary classification response generation tasks.

Finally, both y and $\hat{y}$ are fed to the binary classifier ϕ_b and the loss to be minimized is computed as

$$\begin{aligned} \mathcal{L}_{select} = &- \log p\left(l = 1 \mid \hat{y}\right) \\ &- \log p\left(l = 0 \mid y\right) \end{aligned} \tag{8}$$

$$\begin{aligned} p\left(l = 1 \mid \hat{y}\right) &= \text{sigmoid}\left(\phi_b(\hat{y})\right) \in \mathbb{R}^1 \\ p\left(l = 0 \mid y\right) &= 1 - \text{sigmoid}\left(\phi_b(y)\right) \in \mathbb{R}^1 \end{aligned} \tag{9}$$

4 Experiments

4.1 Dataset

In our workflow, we use the large-scale TOD dataset MultiWOZ 2.1 (Eric et al., 2019) for benchmarks and comparisons with baselines. We follow the preprocessing techniques from (Zhang et al., 2020; Lee, 2021) to replace the specific slot values with placeholders. Table 1 presents more in-depth details and statistics on the contents of the dataset.

4.2 Training procedure

Train/development/test sets are generated with 80%/10%/10% of the samples. We optimize the objectives from section 3 for 15 epochs and report the results from the best performing checkpoint on the development set. In our experiments, we tested different learning rate schedule strategies and found the best results to be achieved with a constant learning rate initialized as 5×10^{-4} with liner warmup for the first 10% of the samples.

For variant 2 of our architecture, a scheduler is used to linearly decrease the Gumbel-Softmax temperature τ with each training iteration. The optimal initial value for τ used to derive the results in Table 2 is 4 and is gradually decreased to 0.8.

4.3 Evaluation Metrics

During inference, the response selection head is not used and the model performs the same way in terms of speed as the T5-small baseline. We compute *BLEU* (Papineni et al., 2002), *Inform* and *Success* metrics for both architecture variants. *Inform* validates whether system entities are correct and *Success* checks whether relevant information is given for all user inquiries. A combined score is derived as 0.5 × (Inform + Success) + BLEU which is consistent with previous studies.

5 Results

5.1 MultiWOZ Benchmarks

Table 2 compares the calculated benchmarks for the two proposed variants of our auxiliary task. As a baseline, we present the results of MTTOD with *T5 base* backbone, which has more than 360 million trainable parameters. In contrast, our models, which use *T5 small* as a backbone, have 102.2 and 105.5 million parameters but achieve higher overall results with total scores of 107.4 and 108.3, respectively.

6 Discussion

Response selection tasks similar to variant 1 of our architecture have been previously applied in models for chit-chatting and question answering (Wolf et al., 2019). For TOD systems such tasks are used in architectures with GPT-2 (AuGPT) and UniLM (GALAXY) backbones resulting in responses with higher text-generation metrics. Our study is the first to provide an in-depth analysis of whether a T5-based model in a task-oriented setting would

Table 1: MultiWOZ dataset statistics

Domain	Train dialogues	Dev dialogues	Test dialogues
Police	245	0	0
Hospital	287	0	0
Attraction	127	11	12
Taxi	326	57	52
Train	282	30	33
Hotel	513	56	67
Restaurant	1199	50	62
Train + Attraction	883	148	163
Hotel + Attraction	437	55	50
Restaurant + Attraction	396	78	70
Restaurant + Train	875	157	155
Restaurant + Hotel	462	59	49
Hotel + Train	1077	149	144
Restaurant + Hotel + Taxi	454	41	42
Restaurant + Attraction + Taxi	431	53	59
Hotel +Attraction + Taxi	444	56	42
Total	8438	1000	1000

Table 2: Benchmark results on MultiWOZ 2.1

Model	Backbone	Selection task	Parameters	Inform	Success	BLEU	Score
MTTOD*	T5 base	None	360.9 M	92.30	84.00	19.41	107.56
MTTOD*	T5 small	None	102.2 M	89.20	80.50	19.14	103.99
RSTOD (ours)	T5 small	After encoder	102.2 M	92.10	83.30	**19.69**	107.39
RSTOD (ours)	T5 small	Differentiable	105.5 M	**93.50**	**84.70**	19.24	**108.34**

* MTTOD benchmarks are reproduced using its public source code. A slight deviation from the results in (Lee, 2021) is caused by a correction in the evaluation scripts as acknowledged on https://github.com/bepoetree/MTTOD.

benefit from selection tasks. The results we present are consistent with related literature since we also observe an increase in generation performance.

Most of the solutions relying on pre-trained language models have big amounts of trainable parameters making them slow to train. In our study, we use a modification of the baseline with T5-small instead of T5-base, reducing the parameters more than 3 times. In variant 1 the shared encoder is responsible for processing more sequences than the baseline - it is slower to train but identical in terms of inference speed and amount of storage space required for its weights. Variant 2 is comparable in terms of train-time and inference-time speed to the baseline but is able to achieve a higher overall score. It employs techniques for overcoming backpropagation issues with the discrete token representations of a generated response sequence[3].

[3]Usually text is generated by picking the most likely tokens from a probability distribution over the token space. This is not a differentiable operation and prevents gradient computations.

7 Future Work

Directions for further research on the topic of TOD systems include testing our proposals on bigger backbone models to evaluate their effectiveness against overfitting, experimenting with additional auxiliary tasks for the current baseline, and introducing data augmentations. Also, whether our classifier heads could be used during inference to perform real-time response selection should be explored.

As a long-term development in the field, we consider various possibilities for building production-ready end-to-end dialogue systems by employing reinforcement learning or semi-supervised learning methods. More experimentally, a generative adversarial network for creative text generation could

also be tested.

8 Conclusion

In this paper, we propose two independent auxiliary tasks for response selection on top of a TOD system transformer baseline. Both tasks demonstrate state-of-the-art results on multiple text generation metrics despite having 3+ times less trainable parameters. The first variant involves a classifier, distinguishing between distractor and ground truth responses, which affects the transformer encoder during training and achieves results consistent with related literature. The second variant applies a novel technique for the TOD problem and involves a classifier, distinguishing between synthetic and ground truth responses. We publish reproducible code implementations of our proposals and present potential directions for future research.

References

Paweł Budzianowski and Ivan Vulić. 2019. Hello, it's gpt-2–how can i help you? towards the use of pre-trained language models for task-oriented dialogue systems. *arXiv preprint arXiv:1907.05774*.

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. *arXiv preprint arXiv:1912.02164*.

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

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *Advances in Neural Information Processing Systems*, 32.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tur. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Shuyang Gao, Sanchit Agarwal, Tagyoung Chung, Di Jin, and Dilek Hakkani-Tur. 2020. From machine reading comprehension to dialogue state tracking: Bridging the gap. *arXiv preprint arXiv:2004.05827*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Claudio Greco, Barbara Plank, Raquel Fernández, and Raffaella Bernardi. 2019. Psycholinguistics meets continual learning: Measuring catastrophic forgetting in visual question answering. *arXiv preprint arXiv:1906.04229*.

Emil Julius Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Wanwei He, Yinpei Dai, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, Luo Si, et al. 2022. Galaxy: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10749–10757.

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

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jonáš Kulhánek, Vojtěch Hudeček, Tomáš Nekvinda, and Ondřej Dušek. 2021. Augpt: Auxiliary tasks and data augmentation for end-to-end dialogue with pre-trained language models. *arXiv preprint arXiv:2102.05126*.

Yohan Lee. 2021. Improving end-to-end task-oriented dialog system with a simple auxiliary task. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1296–1303, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. Mintl: Minimalist transfer learning for task-oriented dialogue systems. *arXiv preprint arXiv:2009.12005*.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

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.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020. Soloist: Few-shot task-oriented dialog with a single pre-trained auto-regressive model. *arXiv preprint arXiv:2005.05298*.

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, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2022. Multi-task pre-training for plug-and-play task-oriented dialogue system.

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*.

Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–929, Online. Association for Computational Linguistics.

Shiquan Yang, Xinting Huang, Jey Han Lau, and Sarah Erfani. 2022. Robust task-oriented dialogue generation with contrastive pre-training and adversarial filtering. *arXiv preprint arXiv:2205.10363*.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14230–14238.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9604–9611.

TopicRefine: Joint Topic Prediction and Dialogue Response Generation for Multi-turn End-to-End Dialogue System

Hongru Wang[1,2,*], **Mingyu Cui**[1,*], **Zimo Zhou**[1], **Kam-Fai Wong**[1,2]
[1]The Chinese University of Hong Kong, Hong Kong, China
[2]MoE Key Laboratory of High Confidence Software Technologies, China
{hrwang,kfwong}@se.cuhk.edu.hk

Abstract

A multi-turn dialogue always follows a specific topic thread, and topic shift at the discourse level occurs naturally as the conversation progresses, necessitating the model's ability to capture different topics and generate topic-aware responses. Previous research has either predicted the topic first and then generated the relevant response, or simply applied the attention mechanism to all topics, ignoring the joint distribution of the topic prediction and response generation models and resulting in uncontrollable and unrelated responses. In this paper, we propose a joint framework with a topic refinement mechanism to learn these two tasks simultaneously. Specifically, we design a three-pass iteration mechanism to generate a coarse response first, then predict corresponding topics, and finally generate a refined response conditioned on predicted topics. Moreover, we utilize GPT2DoubleHeads and BERT for the topic prediction task respectively, aiming to investigate the effects of joint learning and the understanding ability of the GPT model. Experimental results demonstrate that our proposed framework achieves new state-of-the-art performance at the response generation task and the great potential understanding capability of the GPT model.

1 Introduction

Natural Language Generation (NLG), is the task of generating language that is coherent and understandable to humans, and has been applied to many downstream tasks such as text summary (Zhang et al., 2019a; Bar-Haim et al., 2020; Cho et al., 2020; Huang et al., 2020; Gholipour Ghalandari and Ifrim, 2020), machine translation (Li et al., 2020; Baziotis et al., 2020; Cheng et al., 2020; Zou et al., 2020), and dialogue response generation (Radford et al., 2019; Zhou et al., 2018b; Tuan et al., 2019; Zhao et al., 2020; Liu et al., 2020a; Wolf et al., 2019).

[*]These authors contributed equally to this work

Recent works in dialogue response generation usually formulate this task as a sequence-to-sequence problem, leading to inconsistent, uncontrollable, and repetitive responses (Ram et al., 2018). Furthermore, each dialogue has its specific goal and each utterance of the dialogue may contain multiple topics, regardless it is an open-domain dialogue or task-oriented dialogue. As shown in *left* part of Figure 1, the patient seeks medical advice from a doctor and informs him of the attributes and symptoms of the specific disease which form the topics of the conversation. Also, some open-domain dialogue systems have specific goals, such as recommendations, education, etc. For example, a conversational agent interacts with a user to recommend some interesting movies (as shown in *right* part of Figure 1). The entire content flow is guided by the topic thread. These various conversational scenarios propose more challenges for the current multi-turn end-to-end dialogue system, necessitating the model's capability to generate a more informative and topic-related response.

Many researchers propose different methods to guide or control the generation of responses conditioned on specific topics. Some representative works consider incorporating topic information into the sequence-to-sequence framework which applies an attention mechanism to all topics (Xing et al., 2017; Dziri et al., 2019). Other works cast this task as a pipeline system, predict the keywords, then capture the topic, and finally retrieve corresponding response (Tang et al., 2019; Zhou et al., 2020). Another line of work focuses on single-turn topic-aware response generation conditioned on previously given topics (Feng et al., 2018; Yang et al., 2019; Huo et al., 2020). All these methods fall short in two ways. Most of these approaches either heavily rely on the non-autoregressive models like BERT (Devlin et al., 2019) to predict topics or utilize the attention mechanism on all pre-defined topics which do not consider the effect of the histor-

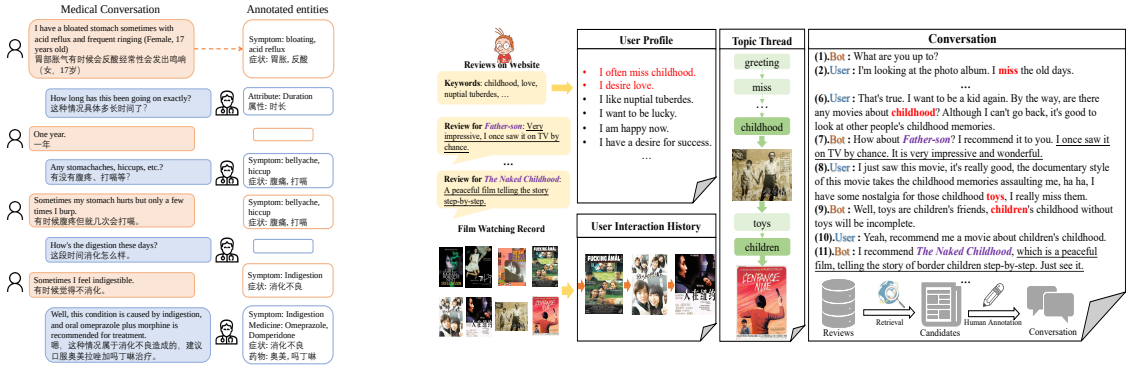


Figure 1: **Left:** MedDG Dataset **Right:** TG-ReDial Dataset. Adapted from (Liu et al., 2020a) and (Liu et al., 2020b) respectively.

ical topic path of multi-turn conversations. Besides that, these works do not model the joint distribution of attribute model $p(a|x)$ and unconditional language model $p(x)$, which is proved effective and powerful (Dathathri et al., 2019).

In this paper, we formulate this problem as a topic-aware dialogue response generation task, aiming to generate informative and topic-related responses that can engage the users. More specifically, we design a three-stage iteration mechanism for the topic-aware response generation task. We generate the coarse response given historical dialogue context and previous topics first, then we require the model to explicitly predict corresponding topics, and then we concatenate the generated coarse response at the first step and the predicted topics at the second step as input to generate a final refined topic-related response. Thus, the model is forced to learn a joint distribution of topics and related responses by optimizing for these three objectives simultaneously.

- We formulate a traditional response generation problem as a topic-aware generation problem and propose a joint framework that can learn topic prediction and dialogue response generation simultaneously.
- We design a topic refine mechanism to control the generation of dialogue response. Our ablation study confirms it can help to generate more informative and topic-related responses, leading to better performance.
- We evaluate our model on two different datasets which consist of two application scenarios: medical auto-diagnosis and conversational recommendation, and we achieve new state-of-the-art performance on both datasets and demonstrate that joint distribution and topic refinement is effective but the understanding ability of GPT2 still needs to be improved.

2 Problem Definition

Given a dialogue $d = \{u^1, u^2, u^3, ..., u^n\}$, a corresponding speaker role path $sr = \{s^1, s^2, s^3, ..., s^n\}$ and its corresponding topic path $tp = \{tw^1, tw^2, tw^3, ..., tw^n\}$ where $s \in R$, $tw \in T$. R and T are pre-defined speaker sets and topic sets. An utterance at ith time step can be expressed by (u^i, s^i, tw^i) which represents the sentence, the speaker, and the topics included in this sentence. tw^i consists of multiple topics or zero topic and each topic is expressed by several words. The problem then can be defined as: given a ith historical dialogue context, speaker role and topic path, $d_i^{n-1} = \{u_i^1, ..., u_i^{n-1}\}, sr_i^{n-1} = \{s_i^1, ..., s_i^{n-1}\}, tp_i^{n-1} = \{tw_i^1, ..., tw_i^{n-1}\}$, find the next topic and generate related responses.

$$y^* = \arg\max_{\theta} p(r^n, tw^n | d^{n-1}, tp^{n-1}, sr^{n-1}) \tag{1}$$

where r^n and tw^n stand for the response and corresponding topics at turn n respectively,. User profile information $p = \{p_1, p_2, ..., p_k\}$ is often provided as additional input, which consists of k sentences to express personal information such as interest. Thus, the objective changes accordingly:

$$y^* = \arg\max_{\theta} p(r^n, tw^n | d^{n-1}, tp^{n-1}, sr^{n-1}, p) \tag{2}$$

Different from other methods, we divide the whole problem into three sub-problems (see the section below). Our objective can be formulated as the following joint distribution:

$$y^* = \arg\max_{\theta} p(r_1^n | d^{n-1}, sr^{n-1}, tp^{n-1}) \\ p(tw^n | d^{n-1}, sr^{n-1}, tp^{n-1}) \\ p(r_2^n | d^{n-1}, sr^{n-1}, tp^{n-1}, (r_1^n, tw^n)) \quad (3)$$

where $p(r_1^n | d^{n-1}, sr^{n-1}, tp^{n-1})$ generate the relatively abbreviated response first, then $p(tw^n | d^{n-1}, sr^{n-1}, tp^{n-1})$ predict the corresponding topics at turn n, and finally, the model refines the abbreviated response r_1^n by maximizing $p(r_2^n | d^{n-1}, sr^{n-1}, tp^{n-1}, (r_1^n, tw^n))$ with the first response r_1^n and corresponding predicted topics tw^n as additional input, which leads to more informative and topic-related response r_2^t.

3 Model

Our model can be divided into three different parts: 1) Stage-One: Response Generation and 2) Topic Prediction; and 3) Stage-Two: Topic Refinement, which corresponds (a), (b), (c) shown in Figure 2 respectively. More details can be checked in the following subsections 3.1, 3.2, and 3.3.

3.1 Stage-One: Response Generation

We formulate the response generation problem using conditional language models e.g. GPT (Radford et al., 2019). Given many dialogues $D = \{d_1, d_2, d_3, ..., d_m\}$, ith dialogue d contains serval training samples $(r^n, tw^n | d^{n-1}, sr^{n-1}, tp^{n-1})$ from different turn n, our objective here is to build a statistical model parameterized by θ to maximize $p_\theta(r^n | d^{n-1}, tp^{n-1}, sr^{n-1})$. Since here we use autoregressive language models to take account of the sequential structure of the response, we need to decompose the joint probability of r^n using the chain rule as follows:

$$p_\theta(r^n | d^{n-1}, tp^{n-1}, sr^{n-1}) = \prod_{t=1}^{T} p_\theta(r_t^n | I) \quad (4)$$

where I stands for $(r_{<t}^n, d^{n-1}, tp^{n-1}, sr^{n-1})$ and $r_{<t}^n$ represents all tokens before t at turn n. The objective of stage one is performed by maximizing the loglikelihood (MLE) of the conditional probabilities in (4) over the entire training dataset:

$$L_{one} = -\sum_{m=1}^{|D|} \sum_{n=1}^{|d|} \sum_{t=1}^{T} log p_\theta(r_t^{m,n} | r_{<t}^{m,n}, \mathcal{H}_m) \quad (5)$$

where $r_{m,n}^t$ is tth token of nth resposne of mth dialogue in training dataset, $\mathcal{H}_m$ represents $(d^{m,n}, tp^{m,n}, sr^{m,n})$ before current response.

3.2 Topic Prediction

Given the historical $\mathcal{H}_m$ of m^{th} dialogue [1], we need not only to generate a suitable response but also to predict the correct topic. Some prior works solve this problem by predicting the topic first and then generating the response (Liu et al., 2020a; Zhou et al., 2020). In this section, different from these works, we propose a framework to jointly learn this task with dialogue response generation task as shown in Figure 2. There are two methods to predict the corresponding topics: (1) BERT-Based Prediction, and (2) GPT-Based Prediction.

3.2.1 BERT-Based Prediction.

Consistent with previous work in text classification (Chen et al., 2019a), we utilize the embedding h_1 of first token $[CLS]$ from BERT (Devlin et al., 2019) to predict the topics, followed by a $softmax$ layer.

$$f(x) = \text{softmax}(W h_1 + b) \quad (6)$$

3.2.2 GPT-Based Prediction.

We adapt GPT2DoubleHeads model (Wolf et al., 2020) to perform the prediction followed (Wolf et al., 2019), since there are two heads: language modeling head and the classification head in the model while the latter one can be used to classify the input dialogue information. Besides that, the shared parameters of GPT may benefit both topic prediction and response generation tasks.

It is noted that there are two types of classification in topic prediction task: *multi-class classification* and *multi-label classification*, owing to the unique characteristic and differences of two datasets: MedDG (Liu et al., 2020a) and TG-ReDial (Liu et al., 2020b). For a *multi-class classification* problem, the global optimization can be reached by minimizing cross-entropy loss defined as follow:

$$L_{topic} = -\sum_{c=1}^{K} y_c log(p_c | \mathcal{H}_m) \quad (7)$$

For a *multi-label classification* problem, it is usually formulated as a sequence of binary decision problems which are optimized by:

[1] It is noted that we do not use $r_{<t}^n$ as input information here.

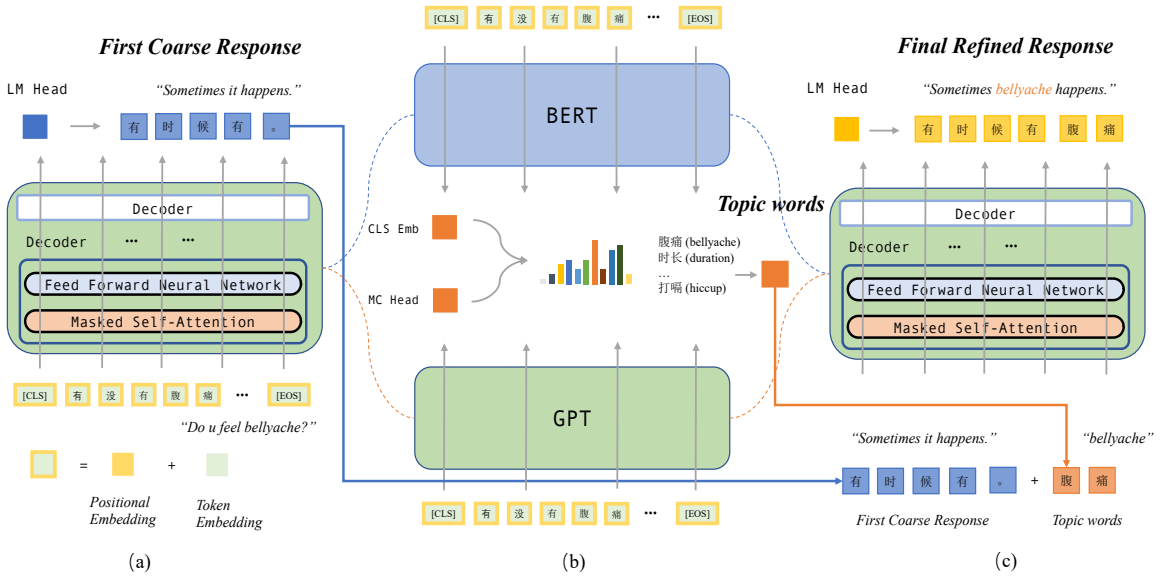


Figure 2: TopicRefine: Joint Framework of Our Proposed Model, which consists of three different modules (a) Stage-One: Response Generation (b) Topic Prediction (c) Stage-Two: Topic Refinement. The (b) module can be implemented by two methods: BERT and GPT, we utilize Stage-One (GPT) and Stage-Two (GPT) to represent the framework with GPT as the backbone for all three modules (orange dashed line), and Stage-Two (BERT) to replace GPT with BERT for (b) module (blue dashed line) in later experiment section.

$$L_{topic} = -\sum_{c=1}^{K} y_c log(p_c|\mathcal{H}_m) + (1-y_c) log(p_c|\mathcal{H}_m) \tag{8}$$

3.3 Stage-Two: Topic Refinement

To generate a more informative and topic-related response, we introduce the *topicRefine* mechanism that refines the generated response condition on the predicted topic [2], as shown in Figure 2 (c).

The refine decoder receives the first generated response r_1^n from the stage-one module and the predicted topic tw^n from the Topic Prediction module as input and outputs a refined response r_2^n. More specifically, we utilize $< topic >$ to indicate the position of topics, so the input can be represented as $\{[CLS], w_r^1, w_r^2, ..., w_r^n, < topic >, w_t^1, w_t^2, ..., w_t^n, < topic >\}$ where $r_1^n = [w_r^1, w_r^2, ..., w_r^n]$, $tw^n = [w_t^1, w_t^2, ..., w_t^n]$. The learning objective is formulated as:

$$L_{refine} = -\sum_{m=1}^{|D|}\sum_{n=1}^{|d|}\sum_{t=1}^{T} log p_\theta(r_t^{m,n}|r_{<t}^{m,n}, \mathcal{H}_m, tw^n) \tag{9}$$

where Eq 9 is similar with Eq 5 except the introduced topic information tw^n here. The parameters are shared by all three modules unless we state otherwise.

[2] If there are k topics predicted by module b, then we simply concatenate all of them together

3.4 Training Objective

The learning objective of our model is the sum of three parts, jointly trained using the "teacher-forcing" algorithm. During training, we feed the ground-truth response only in stage-one and stage-two and minimize the following objective.

$$L_{model} = L_{one} + L_{topic} + L_{refine} \tag{10}$$

At test time, we choose the predicted word by $y^* = argmax_y p(y|x)$ at each time step, and we use greedy search to generate both the response and refined response.

4 Experiment

In this section, we will introduce datasets and baselines first, and then presents implementation details and evaluation metrics of our proposed framework.

4.1 Datasets

MedDG (Liu et al., 2020a) A large-scale high-quality medical dialogue dataset that contains 12 types of common diseases, more than 17k conversation, and 160 different topics consisting of symptoms and attributes. Noted the topic-prediction task here is a multi-label classification problem.

TG-ReDial (Zhou et al., 2020) consists of 10000 two-party dialogues between the user and a recommender in the movie domain which explicitly

incorporates topic paths to enforce natural semantic transitions towards recommendation scenario. For topic-prediction task here, it is a multi-class classification problem. The details of these two datasets can be found in Table 1.

Dataset	MedDG	TG-ReDial
Task Domain	Task-oriented	Recommendation
Language	Chinese	Chinese
Classification Type	Multi-Label	Multi-Class
Dialogue Domain	Medical	Movie
♯ Dialogues	17864	10000
♯ Utterances	385951	129392
♯ Topics	160	2571

Table 1: Statistics of Two Datasets

4.2 Baselines

Seq2Seq. (Sutskever et al., 2014) is a classical attention-based sequence-to-sequence model which builds on top of vanilla RNN encoder and decoder.
HRED. (Serban et al., 2016) extends the traditional RNN encoder by stacking two RNNs in a hierarchical way: one at the word level and one at the utterance level. It is frequently used as a dialogue encoder.
GPT2. (Radford et al., 2019) is a strong baseline for response generation task which demonstrates powerful performance in many related works. It is noted all three methods mentioned above can utilize topic information as additional input which concatenates with utterance in the dialogue. We use **Seq2Seq-Topic**, **HRED-Topic** and **GPT-Topic** to represent these methods respectively.
Redial (Li et al., 2018) is proposed especially for conversational recommendation systems by utilizing an auto-encoder for the recommendation.
KBRD (Chen et al., 2019b) stands for Knowledge-Based Recommendaer Dialog System, which combines the advantages of recommendation system and dialogue generation system.
Transformer (Vaswani et al., 2017) applies a Transformer-based encoder-decoder framework to generate proper responses.
TG-RG (Zhou et al., 2020) is current state-of-the-art method comes with the release of dataset. It predicts the topic first and then generates the response.

4.3 Variants of Our Framework

GPT2DH. The method removes the refinement stage from our framework and jointly trains the response generation and topic prediction tasks (i.e. a and b module in Figure 2) based on the GPT2DoubleHeads model. In this way, the training objective changes to $L_{model} = L_{one} + L_{topic}$ without L_{refine}. We called this method GPT2DH to represent GPT2DoubleHeads (Wolf et al., 2020) which have two heads for classification and generation respectively.

Stage-One (GPT) and Stage-Two (GPT). As shown in Figure 2, this variant represents all three components are implemented by GPT2DoubleHeads model, while **Stage-One (GPT)** represents the first generated response r_1^n and **Stage-Two (GPT)** represents the refined response r_2^n in Equation (3).

Stage-Two (BERT). We replace GPT with BERT only for (b) module in Figure 2. The variant is designed for poor understanding capability of GPT model which leads to noisy predicted topic.

4.4 Implementation Details

We use the same settings for these two datasets. The learning rate is set as 1.5e-4, repetition penalty as 1.0, batch size as 4, warmup steps as 2000, except max context length as 500, max decode length as 50, epochs as 20 for TG-ReDial, max context length as 600, max decode length as 100, epochs as 10 for MedDG. We use `ADAMW` (Loshchilov and Hutter, 2019) to train the model. We emphasize that the role path information is missing in the test data of MedDG. Thus we only use dialogue and topic information in the experiment to keep consistent with test data. It is important to note that our methods do not pre-train on any other big corpus, we just load the parameters provided by (Wolf et al., 2020) and directly fine-tune on the target dataset.

4.5 Evaluation Metrics

For the sake of fair comparison, we adopt the same evaluation metrics as the original two papers (Liu et al., 2020a) and (Zhou et al., 2020). For MedDG, we report BLEU-1, BLEU-4, and Topic-F1 for response generation task, and Precision, Recall, and F1 score for the topic prediction task. For TG-ReDial, we calculate BLEU-1, BLEU2, and BLEU3 for generation and Hit@1, Hit@3, Hit@5 for prediction. It is noted that Topic-F1 requires the topic words appears exactly in the generated response at MedDG dataset.

5 Result and Analysis

In this section, we evaluated the proposed TopicRefine framework at two datasets MedDG and TG-ReDial respectively. And then we further investigate the effects of different response lengths and provide an analysis of human evaluation for dialogue response generation task. At the last, we also investigate the understanding capability of the GPT model in these two datasets.

5.1 Main Result

Model	BLEU-1	BLEU-4	Topic-F1	Avg
Seq2Seq	26.12	14.21	12.63	17.65
Seq2Seq-Topic	35.24	19.20	16.73	23.72
HRED	31.56	17.28	12.18	20.34
HRED-Topic	38.66	21.19	16.58	25.48
GPT2	29.35	14.47	9.17	17.66
GPT2-Topic	30.87	16.56	17.08	21.50
Stage-Two (GPT)	**45.12**	**27.49**	5.40	26.00
Stage-Two (BERT)	44.49	24.62	**17.94**	**29.02**
Stage-One (GPT)	43.86	24.62	11.36	26.61
GPT2DH	43.93	24.35	11.91	26.73

Table 2: Dialogue response generation at MedDG dataset. It is notes that "-Topic" methods use the ground truth topic information in the dataset.

Model	BLEU-1	BLEU-2	BLEU-3
Redial	0.177	0.028	0.006
KBRD	0.223	0.028	0.009
Transformer	0.283	0.068	0.033
GPT2-Topic	0.278	0.064	0.031
TG-RG	0.282	0.067	0.033
Stage-Two (GPT)	0.293	0.085	0.042
Stage-Two (BERT)	**0.294**	**0.086**	**0.043**
Stage-One (GPT)	0.284	0.082	0.041
GPT2DH	0.288	0.086	0.041

Table 3: Recommendation Response Generation at TG-ReDial dataset. It is notes that "-Topic" methods use the ground truth topic information in the dataset.

Table 2 and Table 3 demonstrates the performance of baselines and our proposed framework in both MedDG and TG-ReDial dataset respectively. Our topicRefine framework outperforms the previous state-of-the-art models at both datasets (i.e. GPT2-Topic model at MedDG and TG-RG model at TG-ReDial). More specifically, Stage-Two (GPT) reaches better BLEU score and Stage-Two (BERT) achieves higher Topic-F1 score at MedDG, owing to the existence of noisy topic in former method. Consistent with MedDG dataset, our method gets better performance no matter in Stage-Two (GPT) or Stage-Two (BERT) as shown in Table 3. BLEU-1, BLEU-2, and BLEU-3 all have been improved by different degrees. Another interesting finding is that when explicitly concatenating topic words with dialogue utterances, the GPT-Topic model achieves a higher topic-f1 score, whereas the Stage-Two (GPT) model achieves a lower topic-f1 score, indicating the effectiveness of simply concatenating topic words and the noisy prediction results by GPT.

5.2 Ablation Study

To further investigate the effectiveness of our proposed framework, we add some variants of our proposed framework (i.e. Stage-One (GPT) and GPT2DH) as ablation study. As shown in Table 2 and Table 3, Stage-One (GPT) and GPT2DH achieve comparable results. On the one hand, compared with previous state-of-the-art models, GPT2DH demonstrate more powerful capability which shows the effectiveness of joint learning by incorporating topic prediction. Besides, any Stage-Two model reaches higher BLEU scores than GPT2DH which demonstrate the effectiveness of refine mechanism (i.e. L_{refine}). On the other hand, Stage-Two (GPT) outperforms Stage-One (GPT) in BLEU score (45.12 vs 43.86) but Topic-F1 score (5.40 vs 11.36). We argue that the model tends to generate more topic-related words instead of a specific topic word in the response. This is reasonable since the model is optimized to generate a more informative and topic-related response rather than a specific word.

5.3 Effects of Response Length

To evaluate the impact of different ground-truth response length, we compare the average BLEU score between our model and previous state-of-the-art model (i.e. GPT2-Topic and TG-RG) in MedDG and TG-ReDial respectively. As shown in Figure 3 and Figure 4, our model reaches better performance when the length of golden response is greater than 20 (occupies about 47.6% and 81.9% of test set respectively). As the golden length increases, our improvements also get boosted, which is more obvious at TG-ReDial dataset.

5.4 Generated Sample

Table **??** (See Appendix due to page limit) given some generated response at both datasets. To sum-

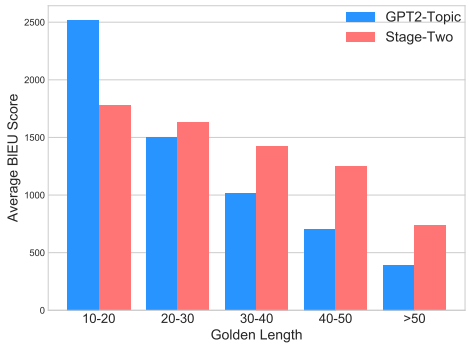


Figure 3: Average BLEU score of MedDG for different golden length

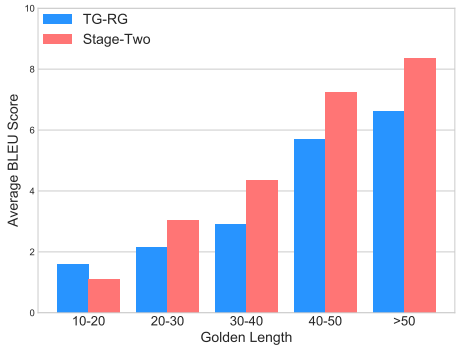


Figure 4: Average BLEU score of TG-ReDial for different golden length

marize, our generated result has the following features:

- For MedDG, since we drop the information of the speaker role path during training and the dialogue between the doctor and the patient is not alternate, some generated responses may represent the perspective of the patient.
- For TG-ReDial, there are some meaningless repeated characters in the result of Stage-One. For example, "。" and "这个电" (this movie) appears twice in response generated by Stage-One. Stage-Two can alleviate this problem by incorporating topic refinement.
- Our Stage-Two model can generate more informative responses conditioned on given topics. Taking the sample of TG-ReDial in Table **??** as an example. For the topic of "memories", the response of ground truth is just a rhetorical question, while the response of our model not only grasps this topic but also recommends one specific movie name related to this topic, which suggests that our model is able to ground multi-turn dialogue generation to some specific topics and tends to be more informative with respect to context.

Model	MedDG		TG-ReDial	
	I	F	I	F
Human	6.99	6.28	7.40	7.28
Baseline	6.18	5.51	6.20	5.69
One	6.32	4.81	6.62	5.66
Two	6.57	6.13	7.30	6.42

Table 4: The result of human evaluation. I and F represent *Information* and *Fluency* respectively. The baseline represents previous sota model *GPT2-Topic* and *TG-RG* in MedDG and TG-ReDial dataset respectively. One represents *Stage-One (GPT)* and Two represents *Stage-Two (GPT)*

5.5 Human Evaluation

To perform human evaluation, we randomly select 50 examples from the outputs of the previous sota model, and our *Stage-One (GPT)* and *State-Two (GPT)* method. The annotators are required to assign two scores for each sentence according to two criteria: (1) information and (2) fluency, ranging from 0 to 10. *information* measures which sentence contains more information (e.g. less repetition). *Fluency* measures which sentence is more proper as a response to a given dialogue context. The evaluation results are calculated by averaging these two scores of all sentences.

Table 4 demonstrates the result of human evaluation. Generally, the score at TG-ReDial dataset is relatively higher than score in MedDG dataset. We attribute this to the MedDG dataset necessitates more expert knowledge and contains many terminologies. Besides that, there is still a large gap between generated response and human response, especially at fluency criteria. In detail, the Stage-One (GPT) performs better than baseline models at information but worse at fluency. Stage-Two (GPT) model gets better scores in both information and fluency criteria than Stage-One (GPT) model and baseline.

5.6 Understanding of GPT Model

Model	P	R	F1
BERT	14.48	**32.95**	20.13
Stage-Two (GPT)	**22.22**	11.16	14.88

Table 5: Result of topic prediction task (multi-label classification) at MedDG dataset

Model	Hit@1	Hit@3	Hit@5
BERT	**0.7651**	**0.8023**	**0.8189**
Stage-Two (GPT)	0.5640	0.7931	0.8122

Table 6: Result of topic prediction task (multi-class classification) at TG-ReDial dataset

Table 5 and Table 6 demonstrate the performance of topic prediction task at MedDG and TG-ReDial datasets respectively. It is obvious that BERT (Devlin et al., 2019) demonstrates more strong understanding ability than GPT (Wolf et al., 2020) model. However, the comparable performance of Hit@3 and Hit@5 between BERT and GPT in Table 6 clearly demonstrates the latter's high understanding potential. The unlocking of potential necessitates a more meticulously designed algorithm or architecture (Dathathri et al., 2019; Liu et al., 2021).

6 Related Work

6.1 Topic-aware Dialogue System

Data-driven, knowledge-grounded dialogue system (Zhou et al., 2018b; Tuan et al., 2019; Zhao et al., 2020) attracts much attention due to the release of large pre-trained language models such as GPT2 (Radford et al., 2019) and DialoGPT (Zhang et al., 2019b). According to different types of knowledge, previous works can be clustered into the following categories: (1) attributes (Zhou et al., 2018a; Zhang et al., 2018a; Xu et al., 2019) (2) persona (Li et al., 2016; Zheng et al., 2019; Wu et al., 2020a; Zhang et al., 2018b) (3) external knowledge graph such as commonsense knowledge (Tuan et al., 2019; Yang et al., 2019; Moon et al., 2019).

Most of previous works for topic-aware dialogue system (Xing et al., 2017; Dziri et al., 2019; Yang et al., 2019; Huo et al., 2020) utilize attention mechanism on all topics at the decode stage to bias the generation probability. (Tang et al., 2019) proposes a structured approach that introduces coarse-grained keywords to control the intended content of system responses and (Xu et al., 2020) proposes Topic-Aware Dual-attention Matching (TADAM) Network to select suitable response but all of their systems are retrieval-based.

6.2 Refine Mechanism

Refine mechanism has been proved to be a effective and compelling technique in both natural language understanding and generation tasks (Zhang et al., 2019a; Wu et al., 2020b; Song et al., 2021). For natural language understanding, (Wu et al., 2020b) design a novel two-pass iteration mechanism to handle the uncoordinated slots problem caused by conditional independence of non-autoregressive model, in which the model utilizes *B-label* output from first phase as input at second phase. For natural language generation, (Zhang et al., 2019a) use refine mechanism to generate refined summary which firstly applies BERT as decoder. Recently, a novel BERT-over-BERT (BoB) model is proposed to solve response generation task and consistency understanding simultaneously (Song et al., 2021). In this paper, we utilize *topicRefine* framwork to build a topic-aware multi-turn end-to-end dialogue system, aiming to generate informative and topic-related dialogue response.

7 Conclusion and Future Work

In this paper, we propose a joint framework with a topic refinement mechanism to solve the topic-aware multi-turn end-to-end dialogue generation problem based on the auto-regressive language model – GPT2 (Wolf et al., 2020). More specifically, we design a three-pass mechanism to jointly learn topic prediction and dialogue response generation tasks, aiming to generate an informative and topic-related response to engage users. Comprehensive experiments demonstrate that our method outperforms previous state-of-the-art models on both MedDG (Liu et al., 2020a) and TG-ReDial (Liu et al., 2020b) datasets, which verifies that the effectiveness of joint learning and refinement mechanism. We will investigate more refined techniques in our future work.

Acknowledgements

This project is supported by a Research Grant, Project ID 4055160, CUHK.

References

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020. From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4029–4039, Online. Association for Computational Linguistics.

Christos Baziotis, Barry Haddow, and Alexandra Birch. 2020. Language model prior for low-resource neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language*

Processing (EMNLP), pages 7622–7634, Online. Association for Computational Linguistics.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019a. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019b. Towards knowledge-based recommender dialog system. 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 1803–1813, Hong Kong, China. Association for Computational Linguistics.

Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020. AdvAug: Robust adversarial augmentation for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5961–5970, Online. Association for Computational Linguistics.

Sangwoo Cho, Kaiqiang Song, Chen Li, Dong Yu, Hassan Foroosh, and Fei Liu. 2020. Better highlighting: Creating sub-sentence summary highlights. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6282–6300, Online. Association for Computational Linguistics.

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. *arXiv preprint arXiv:1912.02164*.

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.

Nouha Dziri, Ehsan Kamalloo, Kory Mathewson, and Osmar Zaiane. 2019. Augmenting neural response generation with context-aware topical attention. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 18–31, Florence, Italy. Association for Computational Linguistics.

Xiaocheng Feng, Ming Liu, Jiahao Liu, Bing Qin, Yibo Sun, and Ting Liu. 2018. Topic-to-essay generation with neural networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4078–4084. ijcai.org.

Demian Gholipour Ghalandari and Georgiana Ifrim. 2020. Examining the state-of-the-art in news timeline summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1322–1334, Online. Association for Computational Linguistics.

Dandan Huang, Leyang Cui, Sen Yang, Guangsheng Bao, Kun Wang, Jun Xie, and Yue Zhang. 2020. What have we achieved on text summarization? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 446–469, Online. Association for Computational Linguistics.

P. Huo, Y. Yang, J. Zhou, C. Chen, and L. He. 2020. Terg: Topic-aware emotional response generation for chatbot. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Bei Li, Ziyang Wang, Hui Liu, Yufan Jiang, Quan Du, Tong Xiao, Huizhen Wang, and Jingbo Zhu. 2020. Shallow-to-deep training for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 995–1005, Online. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 994–1003, Berlin, Germany. Association for Computational Linguistics.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9748–9758.

Wenge Liu, Jianheng Tang, Jinghui Qin, Lin Xu, Zhen Li, and Xiaodan Liang. 2020a. Meddg: A large-scale medical consultation dataset for building medical dialogue system.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too.

Zeming Liu, Haifeng Wang, Zheng-Yu Niu, Hua Wu, Wanxiang Che, and Ting Liu. 2020b. Towards conversational recommendation over multi-type dialogs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1036–1049, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 845–854, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric 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.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 3776–3784. AAAI Press.

Haoyu Song, Yan Wang, Kaiyan Zhang, Wei-Nan Zhang, and Ting Liu. 2021. BoB: BERT over BERT for training persona-based dialogue models from limited personalized data. 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 167–177, 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 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Jianheng Tang, Tiancheng Zhao, Chenyan Xiong, Xiaodan Liang, Eric Xing, and Zhiting Hu. 2019. Target-guided open-domain conversation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5624–5634, Florence, Italy. Association for Computational Linguistics.

Yi-Lin Tuan, Yun-Nung Chen, and Hung-yi Lee. 2019. DyKgChat: Benchmarking dialogue generation grounding on dynamic knowledge graphs. 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 1855–1865, Hong Kong, China. 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. *arXiv preprint arXiv:1706.03762*.

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.

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*.

Bowen Wu, MengYuan Li, Zongsheng Wang, Yifu Chen, Derek F. Wong, Qihang Feng, Junhong Huang, and Baoxun Wang. 2020a. Guiding variational response generator to exploit persona. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 53–65, Online. Association for Computational Linguistics.

Di Wu, Liang Ding, Fan Lu, and Jian Xie. 2020b. SlotRefine: A fast non-autoregressive model for joint intent detection and slot filling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1932–1937, Online. Association for Computational Linguistics.

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 Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3351–3357. AAAI Press.

Can Xu, Wei Wu, Chongyang Tao, Huang Hu, Matt Schuerman, and Ying Wang. 2019. Neural response generation with meta-words. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5416–5426, Florence, Italy. Association for Computational Linguistics.

Yi Xu, Hai Zhao, and Zhuosheng Zhang. 2020. Topic-aware multi-turn dialogue modeling. *arXiv preprint arXiv:2009.12539*.

Pengcheng Yang, Lei Li, Fuli Luo, Tianyu Liu, and Xu Sun. 2019. Enhancing topic-to-essay generation with external commonsense knowledge. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2002–2012, Florence, Italy. Association for Computational Linguistics.

Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. 2019a. Pretraining-based natural language generation for text summarization. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 789–797, Hong Kong, China. Association for Computational Linguistics.

Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2018a. Learning to control the specificity in neural response generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1108–1117, Melbourne, Australia. Association for Computational Linguistics.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018b. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019b. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*.

Xueliang Zhao, Wei Wu, Can Xu, Chongyang Tao, Dongyan Zhao, and Rui Yan. 2020. Knowledge-grounded dialogue generation with pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3377–3390, Online. Association for Computational Linguistics.

Yinhe Zheng, Rongsheng Zhang, Xiaoxi Mao, and Minlie Huang. 2019. A pre-training based personalized dialogue generation model with persona-sparse data.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018a. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018b. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4623–4629. ijcai.org.

Kun Zhou, Yuanhang Zhou, Wayne Xin Zhao, Xiaoke Wang, and Ji-Rong Wen. 2020. Towards topic-guided conversational recommender system. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4128–4139, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Wei Zou, Shujian Huang, Jun Xie, Xinyu Dai, and Jiajun Chen. 2020. A reinforced generation of adversarial examples for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3486–3497, Online. Association for Computational Linguistics.

Prior Omission of Dissimilar Source Domain(s) for Cost-Effective Few-Shot Learning

Zezhong Wang[1,2,*], **Hongru Wang**[1,2,*], **Wai-Chung Kwan**[1,2], **Kam-Fai Wong**[1,2]
[1]The Chinese University of Hong Kong, Hong Kong, China
[2]MoE Key Laboratory of High Confidence Software Technologies, China
[1,2]{zzwang, hrwang, wckwan, kfwong}@se.cuhk.edu.hk

Abstract

Few-shot slot tagging is an emerging research topic in Natural Language Understanding (NLU). Conventional few-shot approaches use all the data from the source domains without considering inter-domain relations and implicitly assume each sample in the domain contributes equally. However, our experiments show that transferring knowledge from dissimilar domains will introduce extra noises that decrease the performance of models. We propose an effective similarity-based method to select data from the source domains to tackle this problem. In addition, we propose a Shared-Private Network (SP-Net) for the few-shot slot tagging task. The words from the same class would have some shared features. We extract those shared features from the limited annotated data on the target domain and merge them as the label embedding to help us predict other unlabelled data on the target domain. The experiment shows that our method outperforms the state-of-the-art approaches with fewer source data. The result also proves that some training data from dissimilar sources are redundant and even negative for the adaptation.

1 Introduction

Slot tagging (Tur and De Mori, 2011), one of the crucial problems in Natural Language Understanding (NLU), aims to recognize pre-defined semantic slots from sentences and usually is regarded as a sequence labeling problem (Sarikaya et al., 2016). For example, given the sentence "Book a ticket to London", the word "London" should be recognized as the slot "CITY" by the NLU model.

Currently, most of the methods for the slot tagging task have a notorious limitation in that they require a lot of annotated data. However, there are almost infinite long tail domains in the real scenarios (Zhu et al., 2014) so it is nearly impossible to annotate sufficient data for each domain. Therefore, few-shot learning methods (Ravi and Larochelle, 2016) have received attention as they can transfer the knowledge learned from the existing domains to new domains quickly with limited data.

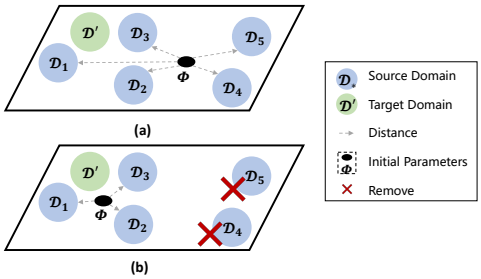


Figure 1: The difference between training with (a) all data and (b) data selection. The dashed line represents the distance among different domains in the parameter space with the centroid (Φ). With data selection, we remove the dissimilar domains $\mathcal{D}_4$ and $\mathcal{D}_5$ from training and the centroid will be closer to the target domain $\mathcal{D}'$.

Current works (Yoon et al., 2019; Liu et al., 2020; Wang et al., 2021) proposed various methods to improve the performance of slot tagging few-shot learning, but most of them focus on "how" to transfer rather than "what" should be transferred. The knowledge from the not-relevant source domain is hard to help the model identify the slots in the new domain. Further, such kind of knowledge is redundant and sometimes could be regarded as noises that even deteriorate the performance (Wang et al., 2019; Meftah et al., 2021). We observe this phenomenon and prove the existence of the negative transfer in the experiment. To this end, we propose a similarity-based method to evaluate the inter-domain relation and indicate which domains should be selected for training. Specifically, we calculate three different similarities, including target vocabulary covered (TVC), TF-IDF similarity (TIS), and label overlap (LO) between domains, and combine them with different weights. The combined similarity function selects data from both corpus level and label level, which is more com-

[*]These authors contributed equally to this work.

prehensive. In this way, the dissimilar sources will be rejected, and the initial parameters of the model will be naturally more closed to the local optimum of the target domain. A high-level intuition of the difference between training with all data and training with data selection is shown in Figure 1.

After selecting pertinent data, we also propose a solution about "how" to transfer knowledge for a few-shot slot tagging task. Specifically, we build a Shared-Private Network to capture stable label representations under the few-shot setting. Many works (Hou et al., 2020; Zhu et al., 2020; Liu et al., 2020) try to enhance the accuracy of slot identification from the label representation engineering. They assign each label with a semantic vector (Snell et al., 2017; Hou et al., 2020; Zhu et al., 2020; Yoon et al., 2019) rather than a simple one-hot encoding. However, the quality of the label representations highly depends on the volume of the training samples and suffers from the unstable problem under the few-shot setting due to the extremely biased data distribution. Hence, we propose the Shared-Private Network separate the shared and private features from the limited samples. The words with the same label share common information. They are extracted and saved as shared features. Other parts are regarded as detailed information related to the words and will be saved as private features. After filtering the detailed information out, the label representation generated according to the shared features will be more robust against the annotation shortage problems in the few-shot setting.

The contributions of this work are as follows:

- We propose a similarity-based method to measure the relation among domains to guide data selection and to avoid negative knowledge transfer in few-shot learning.
- We propose the Shared-Private Network to extract more stable label representation with limited annotations.
- We prove the existence of negative transfer via experiments and give explanations about this phenomenon via visualization.

2 Related Work

Convention studies in slot tagging mainly focus on proposing and utilizing deep neural networks to recognize the semantic slots in given contexts (Shi et al., 2016; Kim et al., 2017). However, most of these models need a large amount of annotated data which is scarce in the real world, especially for those minority domains. Recent works (Bapna et al., 2017; Shah et al., 2019; Rastogi et al., 2019; Liu et al., 2020) propose several few-shot learning methods for slot tagging and developed domain-specific model with limited annotated data. It is worth noting that, due to the lack of annotation on the target domain, both approaches paid attention to label representation engineering rather than using conventional one-hot encoding directly. But building label representation with limited annotations is still a challenge. To stabilize the effectiveness of label representation, we proposed a Shared-Private network to learn representation from shared information of words.

Besides that, negative transfer that transferring knowledge from the source can have a negative impact on the target has been founded in many tasks (Wang et al., 2019; Chen et al., 2019; Gui et al., 2018). Because of this phenomenon, recent methods for relation analysis between source and target domains have been proposed. Gururangan et al. (2020) use vocabulary overlap as the similarity between two datasets and emphasize the significant impact of domain-adaptive for pre-training. Dai et al. (2019) study different similarity methods, including target vocabulary covered (TVC), language model perplexity (PPL), and word vector variance (WVV) to select data for pre-training tasks. However, a single similarity function does not work well in the few-shot setting. Different similarity methods always give diverse data selection strategies and are hardly consistent. To this end, we propose a comprehensive indicator that combines three similarity functions to guide the data selection in the few-shot setting.

3 Problem Definition

We follow the same task definition as Hou et al. (2020). Given a sentence $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ as a sequence of words, slot tagging task aims to assign the corresponding label series $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ to indicate which classes the words should belong to. A domain $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N_\mathcal{D}}$ is a set of $(\mathbf{x}, \mathbf{y})$ pairs that from same scenario and $N_\mathcal{D}$ is the number of sentences in domain $\mathcal{D}$.

In few-shot setting, models are trained from source domain $\{\mathcal{D}_1, \mathcal{D}_2, \cdots\}$ and are applied to the target domain $\{\mathcal{D}'_1, \mathcal{D}'_2, \cdots\}$ which are new to the models. It is worth note that there are only few labeled samples, which make up the support set

$\mathcal{S} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N_S}$, in each target domain $\mathcal{D}'_j$. For each unique N labels (N-way) in support set $\mathcal{S}$, there are K annotated samples (K-shot). Besides that, the samples in the target domain $\mathcal{D}'_j$ are unlabeled.

Thus, few-shot slot tagging task is defined as follows: given a K-shot support set $\mathcal{S}$ and a query sentence $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, determine the corresponding labels sequence $\mathbf{y}^*$:

$$\mathbf{y}^* = (y_1^*, y_2^*, \cdots, y_n^*) = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}, \mathcal{S}) \quad (1)$$

4 Data Selection

In this section, we first show the existence of negative knowledge transfer among domains. The phenomenon demonstrates the necessity of data selection. Then introduce our similarity-based data selection strategy that can be used to avoid negative knowledge transfer to improve performance in few-shot slot tagging.

4.1 Negative Knowledge Transfer

Due to negative knowledge transfer, some knowledge the model learned before is useless and may affect the judgment of the model on the new domains, which will degrade the performance. In the preliminary study, we train the model with all different combinations of source domains and record their performance. The relation between the number of source domains and their corresponding performance is shown in Figure 2. Overall, with more training domains, the performance would be better. However, comparing the maximum values, it is evident that training with three source domains outperforms training with 4. This phenomenon indicates that more source domains may even decrease the performance and proves the existence of negative knowledge transfer. It also inspires us that the model will achieve a better result with proper data selection.

4.2 Selection Strategy

An indicator is needed to select data or source domains before training to avoid negative knowledge transfer. Given a group of data from source domain and the data of target domain, the indicator should output a score that can reflect how fit are these source data for transferring knowledge to the target. Ideally, the indicator score behaves linearly with the performance so that a higher indicator score can lead to better performance. In this way, the group of source data with the highest indicator score can be selected as the best choice for training.

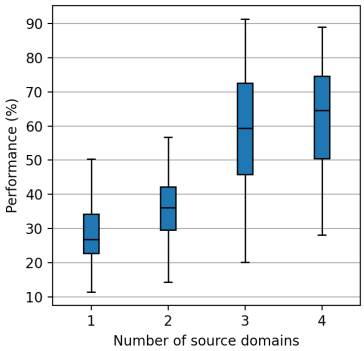


Figure 2: The relationship between performance (y-axis), specifically the F1 score, and the number of source domains (x-axis).

The data that can be leveraged includes the source domains $\{\mathcal{D}_1, \cdots, \mathcal{D}_M\}$ with sufficient labels, the support set $\mathcal{S}_j$ with labels in the target domain $\mathcal{D}'_j$, and the query set $\mathcal{Q}_j$ without labels. Notice that the data in the support set $\mathcal{S}_j$ is much less than the query set $\mathcal{Q}_j$. Considering the attributes mentioned above and the data we can use, we investigate three similarity functions as indicators for data selection.

Target Vocabulary Covered (TVC) is a significant corpus level feature that represents the overlap of vocabulary between source domain(s) and a target domain and is defined as:

$$\mathbf{TVC}(\mathcal{D}_i, \mathcal{D}'_j) = \frac{\left|V_{\mathcal{D}_i} \cap V_{\mathcal{D}'_j}\right|}{\left|V_{\mathcal{D}'_j}\right|} \quad (2)$$

where $V_{\mathcal{D}_i}$ and $V_{\mathcal{D}'_j}$ are the vocabularies (sets of unique tokens) of the source domain $\mathcal{D}_i$ and the target domain $\mathcal{D}'_j$ respectively and $|\cdot|$ is the norm operation that indicates the size of the set. Intuitively, if most of words in the target domain have already appeared in the sources, the word embeddings should have been well trained so that improves the performance.

TF-IDF Similarity (TIS) is another corpus level feature (Bao et al., 2020). We treat each domain as a document and calculate their `tf-idf` features (Salton and Buckley, 1988; Wu et al., 2008). Cosine similarity is used to evaluate the correlation between the sources and the target. Compared with TVC, TIS assigns each word a weight according to the term frequency and inverse document frequency, which takes fine-grained corpus feature

into account. The details are shown below:

$$\text{tf}_{i,j} = \frac{n_{ij}}{\sum_k n_{k,j}} \tag{3}$$

where n_{ij} is the times of word t_i appeared in domain $\mathcal{D}_j$.

$$\text{idf}_i = \lg\left(\frac{M}{|\{j : t_i \in \mathcal{D}_j\}_{j=1}^{M}|}\right) \tag{4}$$

where M is the total number of domains. And the `tf-idf` feature is the product of `tf` and `idf`:

$$\text{tf-idf}_j = \text{tf}_{i,j} \cdot \text{idf}_i \tag{5}$$

tf-idf$_j$ can be regarded as the word distribution feature of the domain j, and cosine similarity is used to evaluate the correlation between the two domains:

$$\text{TIS}(\mathcal{D}_i, \mathcal{D}_j) = \frac{\text{tfidf}_{\mathcal{D}_i} \cdot \text{tfidf}_{\mathcal{D}_j}}{||\text{tfidf}_{\mathcal{D}_i}||_2 \cdot ||\text{tfidf}_{\mathcal{D}_j}||_2} \tag{6}$$

where $||\cdot||_2$ is the Euclidean norm.

Label Overlap (LO) is a label level feature that represents the overlap of labels between source domains and the target domain. Although labels are scarce in the target domain under the few-shot setting, the types of labels are not. Every label on the target domain at least appeared K times (K-shot) in the support set $\mathcal{S}$ and therefore, the types of the labels are complete. Hence, label overlap is also a good choice as a data selection indicator:

$$\text{LO}(Y_i, Y_j) = \frac{|Y_i \cap Y_j|}{|Y_j|} \tag{7}$$

where Y_i and Y_j stand for the unique label set of the source domain $\mathcal{D}_i$ and the target domain $\mathcal{D}'_j$, respectively.

Each similarity function only focus on a single aspect, i.e. the corpus level information or the label level. Therefore, it is inevitable to introduce bias when we select data with them. Naturally, we come up with a strategy that combines all three similarity scores as the indicator to give a more stable guidance for data selection. Assume that one of the combinations, i.e. $C_{\theta_1,\theta_2,\theta_3}(\text{TVC}_i, \text{TIS}_i, \text{LO}_i) = \theta_1\text{TVC}_i + \theta_2\text{TIS}_i + \theta_3\text{LO}_i$, is linear with the performance, our goal is to find the best value of θ_1, θ_2, and θ_3. For a better reading experience, $C_{\theta_1,\theta_2,\theta_3}(\text{TVC}_i, \text{TIS}_i, \text{LO}_i)$ is abbreviated to C_i.

Algorithm 1 Training with combination of source domains

Require: Set of source domains $\{\mathcal{D}_1, \cdots, \mathcal{D}_M\}$; Target domain $\mathcal{D}'$; Model $\mathcal{F}$;

for $1 \le i \le M$ **do**
  all_combination = $combination(\{\mathcal{D}_1, \cdots, \mathcal{D}_M\}, i)$
  // Select i domain(s) from M for training.
  for $1 \le j \le |\text{all\_combination}| - 1$ **do**
    combination = all_combination$[j]$
    // e.g. combination = $[\mathcal{D}_1, \mathcal{D}_3]$
    $\mathcal{D}_{\text{training}} \leftarrow Merge(\text{combination})$
    TVC = $TVC(\mathcal{D}_{\text{training}}, \mathcal{D}')$
    TIS = $TIS(\mathcal{D}_{\text{training}}, \mathcal{D}')$
    LO = $LO(\mathcal{D}_{\text{training}}, \mathcal{D}')$
    train $\left(\mathcal{F}(\mathcal{D}_{\text{training}})\right)$ until Loss converge
    $\hat{p}_i$ = eval$((\mathcal{F}(\mathcal{D}'))$
  end for
end for

Following the least squares method (Merriman, 1877), we design the objective function as follows:

$$\begin{aligned} &\arg\min_{\theta_1,\theta_2,\theta_3,w,b} \frac{1}{N_E}\sum_{i=1}^{N_E} \|[wC_i + b] - \hat{p}_i\|^2 \\ &\text{s.t.} \quad w > 0, b \ge 0 \end{aligned} \tag{8}$$

where w and b are the weight and bias of the linear function to simulate the linear relation between the indicator score and the performance. N_E is the number of the experiments, and $\hat{p}_i$ is the actual performance of the experiment i. TVC$_i$, TIS$_i$, and LO$_i$ are the TVC score, TIS score, and LO score between the source domains and the target domain in the experiment i.

To solve the problem in equation (8), we design a scheme to generate samples with the combination of source domains. We pre-define the number of source domains and enumerate all combinations. The three similarity scores between the combination of source domains and target domains will be calculated and recorded. Then we train the model with the combination and record the final performance on the target domain. In this way, we get sufficient tuples $(\texttt{TVC}, \texttt{TIS}, \texttt{LO}, p)$ to figure out the optimum θ_1, θ_2, and θ_3 (see Algorithm 1).

With sufficient samples, we fit them with the linear function in equation (8) and optimize w, b, θ_1, θ_2, and θ_3 via `SGD` (Curry, 1944). Due to the data distribution bias of different domains, we finally assign different w_j and b_j for each target domain $\mathcal{D}'_j$ to acquire a better linear relation. For the combination weights θ_1, θ_2, and θ_3, we keep same for different target domains. Further, we still have the following points to declare:

- The parameters w and b are learnable but unnecessary for data selection. They are not a part of the indicator and are only used to observe the linear relation between the combination similarity scores and the corresponding performance.
- Due to the cross-validation setting in the real dataset (e.g., SNIPS), to avoid data leakage of the target domain, we obtain θ_1, θ_2, and θ_3 according to the validation domain for each target. The combination from the validation domain still works well on the target and can prove the generality of this strategy.
- Although training with a combination of source domains is time-consuming, it can be adapted to different domains once the optimum combination weights have been found.

After that, we can select domains according to the optimum w^*, b^*, θ_1^*, θ_2^*, and θ_3^*. The domains which can achieve a higher combined similarity score may lead to better performance, and this can be formulated as:

$$\arg\max_i \quad w^* \left(\theta_1^* \text{TVC}_i + \theta_2^* \text{TIS}_i + \theta_3^* \text{LO}_i\right) + b^* \tag{9}$$

And due to $w > 0$, equation (9) is equivalent to:

$$\arg\max_i \quad \theta_1^* \text{TVC}_i + \theta_2^* \text{TIS}_i + \theta_3^* \text{LO}_i \tag{10}$$

In this way, the domain specific w and b are eliminated.

5 Shared-Private Network

Based on the Prototypical Network (Snell et al., 2017), we propose the Shared-Private Network (SP-Net) to gain more representative label embeddings. The workflow is divided into two stages: SP-Net extracts label embeddings for each class from the support set in the first stage. SP-Net predicts each query sentence in the second stage according to the label embeddings extracted from stage one. Figure 3 illustrates this process.

(a) Encode Firstly, sentences are encoded into word embeddings via BERT (Devlin et al., 2019). Given a sentence $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ as a sequence of words, BERT will generate their corresponding contextual word embeddings $\mathbf{E} = (E_1, E_2, \cdots, E_n)$, where $E_i \in \mathbb{R}^h$. h is the hidden size of the word embedings.

(b) Extract shared features Although words are different, there is common information among words from the same class. Intuitively, the same class words always appear in a similar context with similar syntax. And in some cases, they can even be replaced with each other without any grammatical mistakes. For example, even though we replace the phrase “Hong Kong" with “New York" in Figure 3, the sentence still makes sense. Common information can help us generate scalable label embeddings that can represent most of the words in a class. The shared layer in the framework is designed for this. In this work, we implement the shared layer with a residual linear function, and the shared feature of a word is calculated as follows:

$$E_i^s = E_i + \text{RELU}(E_i W_s + b_s) \tag{11}$$

where $W_s \in \mathbb{R}^{h \times h}$ and $b_s \in \mathbb{R}^h$ are the weight and bias of the shared layer, respectively. RELU is the rectified linear unit function (Maas et al., 2013).

(c) Extract private features Besides the shared information, each word still has its specific information. Recalling the phrase replacing case mentioned in Figure 3, although the sentence is without any grammatical mistakes after phrase replacing, the meaning has been changed. This is due to the private information carried by the word. The private information is ineffective and can be harmful to label embeddings as they lack generality. Less private information can lead to a better quality of label embeddings, and therefore, the private layer is designed to extract private information from the word embeddings. The private layer is also implemented with a residual linear function, and the private feature of a word is calculated as follows:

$$E_i^p = E_i + \text{RELU}(E_i W_p + b_p) \tag{12}$$

where $W_p \in \mathbb{R}^{h \times h}$ and $b_p \in \mathbb{R}^h$ is the weight and bias of the private layer, respectively. So far, the shared layer and private layer are symmetrical and share the same design.

(d) orthogonality constrain To ensure the shared features and private features are separated completely, we introduce the following constraints:

- The shared features of the words in the same class should be close to each other.
- The private features of words should be diverse even though they belong to the same class.
- The shared and private features of a word should not overlap.

For the first requirement, Chen et al. (2020) proposed using contrastive loss that can make the same

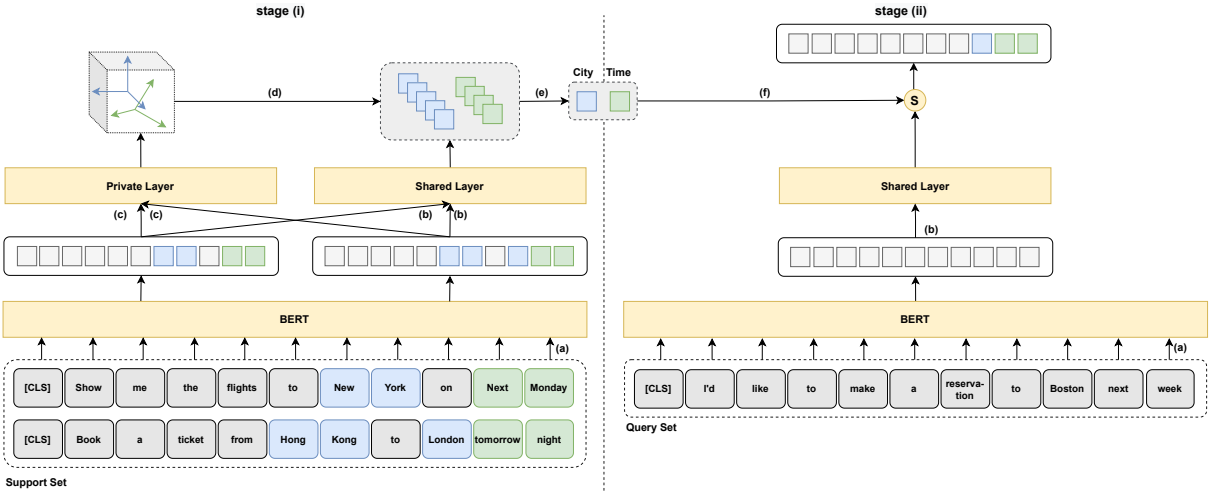


Figure 3: This is the workflow of SP-Net. In this case, the support set contains 2 sentences, and the query set contains 1. The details of processes (a) encode, (b) extract shared features, (c) extract private features, (d) orthogonality constrain, (e) extract label embeddings, and (f) predict are introduced in the main body.

samples close and different samples far apart. The similarity between samples are defined as follows:

$$\text{sim}(E_i^s, E_j^s) = \frac{E_i^{s\top} E_j^s}{\|E_i^s\| \|E_j^s\|} \tag{13}$$

The loss in the first requirement is defined as:

$$\mathcal{L}_1 = \mathop{\mathbb{E}}_{c} \left[-\log \frac{\sum\limits_{\{i;y_i=c\}} \sum\limits_{\{j;y_j=c\}} e^{\text{sim}(E_i^s, E_j^s)/\tau}}{\sum\limits_{\{i;i\in\mathcal{S}\}} \sum\limits_{\{j;j\in\mathcal{S}\}} e^{\text{sim}(E_i^s, E_j^s)/\tau}} \right] \tag{14}$$

where τ is the temperature parameter and c is the class. The numerator is the sum of the similarity scores whose class is c. The denominator is the sum of all the similarity scores. Specifically, embeddings in the same class present a high similarity score and the numerator is large, and the loss decreases.

For the second requirement, according to the covariance of two variables, we define the divergence between two embeddings as:

$$D(E_i^p, E_j^p) = (E_i^p - \mathbb{E}^p)^T (E_j^p - \mathbb{E}^p) \tag{15}$$

where $\mathbb{E}^p$ is the mean vector of all private embeddings in the set. The loss in the second requirement is:

$$\mathcal{L}_2 = -\frac{1}{|\mathcal{S}|^2} \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{S}} \log D(E_i^p, E_j^p) \tag{16}$$

where $|\mathcal{S}|$ is the size of the support set, i.e., the number of words. Higher divergence among the private embeddings will lead to lower loss. We also implement `L2-norm` to restrain the increase of the parameters.

The third requirement refines the shared features further. We introduce the orthogonality constraints (Liu et al., 2017) to force the shared embedding independent of the private embedding:

$$\mathcal{L}_3 = \frac{1}{|\mathcal{S}|} \sum_{i\in\mathcal{S}} \left\| E_i^{s\top} E_i^p \right\|_2 \tag{17}$$

where $\|\cdot\|_2$ is the Euclidean norm.

(e) Extract label embeddings Label embeddings are extracted from shared embeddings for each class. We take the mean vector of the shared embeddings which belong to class c as the label embedding:

$$E^c = \frac{1}{|\{y_i = c\}|} \sum_{\{y_i=c\}} E_i^s \tag{18}$$

where E^c is the label embedding of the class c.

(f) Predict We calculate the similarity between shared embeddings of the query sentence with the label embeddings. We provide various options, and here we take cosine similarity as an example:

$$p_i^c = \frac{E_i^{s\top} E^c}{\|E^s\| \|E^c\|} \tag{19}$$

where p_i^c is the similarity between word i with class c and can also be regarded as the confidence that the word belongs to this class. The class with the highest similarity will be considered as the prediction for the word. We take the binary cross-

	Model	We	Mu	Pl	Bo	Se	Re	Cr	Avg.
1-shot	SimBERT	36.10	37.08	35.11	68.09	41.61	42.82	23.91	40.67
	TransferBERT	55.82	38.01	45.65	31.63	21.96	41.79	38.53	39.06
	L-TapNet+CDT+PWE (Hou et al., 2020)	71.53	60.56	66.27	**84.54**	76.27	70.79	62.89	70.41
	L-ProtoNet+CDT+VPB (Zhu et al., 2020)	73.12	57.86	69.01	82.49	75.11	73.34	70.46	71.63
	BERT-ProtoNet	60.01	43.33	52.42	44.37	47.86	50.91	39.04	45.65
	SP-Net	70.67	59.27	69.58	82.80	76.92	72.49	74.63	72.34
	SP-Net + Domain Selection	**76.07**	**64.29**	**71.10**	84.19	**81.63**	**73.66**	**76.41**	**75.34** (+3.71)
5-shot	SimBERT	53.46	54.13	42.81	75.54	57.10	55.30	32.38	52.96
	TransferBERT	59.41	42.00	46.07	20.74	28.20	67.75	58.61	46.11
	L-TapNet+CDT+PWE(Hou et al., 2020)	71.64	67.16	75.88	84.38	82.58	70.05	73.41	75.01
	L-ProtoNet+CDT+VPB(Zhu et al., 2020)	82.93	69.62	80.86	91.19	86.58	**81.97**	76.02	81.31
	BERT-ProtoNet	68.98	59.31	62.42	81.35	78.91	67.57	71.69	70.03
	SP-Net	83.92	69.37	79.47	89.43	87.95	77.75	80.31	81.17
	SP-Net + Domain Selection	**84.03**	**71.09**	**82.01**	**92.13**	**89.44**	80.71	**80.88**	**82.90** (+1.59)

Table 1: F1 scores of few-shot slot tagging on SNIPS dataset.

Model	1-shot					5-shot				
	News	Wiki	Social	Mixed	Avg.	News	Wiki	Social	Mixed	Avg.
SimBERT	19.22	6.91	5.18	13.99	11.32	32.01	10.63	8.20	21.14	18.00
TransferBERT	4.75	0.57	2.71	3.46	2.87	15.36	3.62	11.08	35.49	16.39
L-TapNet+CDT+PWE	**44.30**	12.04	20.80	15.17	23.08	45.35	11.65	23.30	20.95	25.31
L-ProtoNet+CDT+VPB	43.47	10.95	**28.43**	33.14	29.00	56.30	18.57	35.42	44.71	38.75
SP-Net	43.95	**13.02**	27.77	**34.05**	**29.70**	**57.70**	18.62	**36.41**	**44.97**	39.42
SP-Net Domain Selection	43.95	13.02	27.77	34.05	29.70 (+0.70)	57.70	**21.11**	36.41	44.97	**40.05** (+1.30)

Table 2: F1 scores of few-shot slot tagging on NER dataset. The performance improvements of SP-Net Domain Selection compared to all baselines are significant (validated by Student's t-test with p-value < 0.01).

entropy loss to measure the error in each class:

$$\mathcal{L}_4 = \frac{1}{|\mathcal{Q}|}\sum_{i}^{|\mathcal{Q}|}\sum_{c}^{C} y_i \log p_i^c + (1-y_i)\log\left(1-p_i^c\right) \tag{20}$$

where C is the number of unique labels in the query set and $|\mathcal{Q}|$ is the number of words in the query set.

Finally, we combine the $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, and $\mathcal{L}_4$ with different weights as the cost function:

$$\mathcal{L} = \alpha\mathcal{L}_1 + \beta\mathcal{L}_2 + \gamma\mathcal{L}_3 + \delta\mathcal{L}_4 \tag{21}$$

where α, β, γ, and δ are hyperparameters determined by the experiments.

6 Experiments

6.1 Dataset

We evaluate the proposed method following the same experiment setting provided by Hou et al. (2020) on SNIPS (Coucke et al., 2018) and NER dataset (Zhu et al., 2020). SNIPS contains seven domains, including Weather (We), Music (Mu), PlayList (Pl), Book (Bo), Search Screen (Se), Restaurant (Re), and Creative Work (Cr), and the sentences in SNIPS are annotated with token-level `BIO` labels for slot tagging. Each domain will be tested in turn following the cross-validation strategy. Five domains are used for training and one for evaluation in each turn. In each domain, the data are split into 100 episodes (Ren et al., 2018). For the sake of fair peer comparison, the selection of evaluation domain and episodes construct are kept the same with Hou et al. (2020). The NER dataset contains four domains: News, Wiki, Social, and Mixed. In addition, because the number of domains in the NER dataset is too short, we randomly split domains into pieces and select those pieces via the combined similarity function. More training details can be found in the appendix.

6.2 Baselines

SimBERT assigns a label to the word according to cosine similarity of word embedding of a fixed BERT.

TransferBERT directly transfers the knowledge from the source domain to the target domain by parameter sharing.

L-TapNet+CDT+PWE (Hou et al., 2020) combines with the label name representation and a special CRF framework.

L-ProtoNet+CDT+VPB (Zhu et al., 2020) utilizes the powerful distance function VPB to boost the performance of the model.

BERT-ProtoNet is the model proposed in this work which is without the Shared-Private layer. This model is used for ablation study.

SP-Net is the Shared-Private Network proposed in this work.

SP-Net + Domain Selection is also SP-Net, but it is trained with the selected data according to the data selection strategy we proposed.

6.3 Main Results

Table 1 shows the results of 1-shot and 5-shot on the SNIPS dataset. Generally speaking, the SP-Net achieves the best performance on the 1-shot setting and comparable performance on the 5-shot setting (0.14% adrift of SOTA). The data selection strategy dramatically enhances the performance on both the 1-shot and 5-shot settings. With the data selection, the performance of SP-Net is far beyond other baselines.

The result on the NER dataset also proves the effectiveness of our method (See Table 2). It is noticed that, due to the short of the data, combined similarity select all data on most domains except Wiki of 5-shot task. Therefore the result of SP-Origin and SP-Domain Selection are nearly the same.

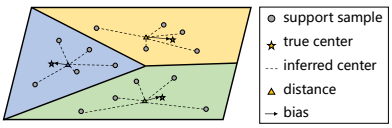


Figure 4: This is diagram shows the automatic correction of distribution bias when the number of supports increased. The circles are samples in the support set and triangles are the inferred center, as well as label embedding, according to the supports. Stars are the true center of classes.

The effect of the Shared-Private Network is more remarkable if the number of support samples is less. The SP-Net outperforms all baseline in the 1-shot setting, but in 5-shot, it achieves comparable performance. The shared-private Network essentially corrects the bias between the label embedding and the center of the class. The bias will be more severe if the support is less. With the increase in the number of supports, bias could be suppressed to some extent (see Figure 4). Some other methods, like label description (Hou et al., 2020), can also correct such kind of bias if enough supports are given. But when the supports are highly scarce, Shared-Private Network performs the best.

6.4 Analysis

We further visualize the relation between the performance with the similarity function and compare combined similarity with TVC in Figure 5. We firstly sample some combinations of source domains and train the model. Then we calculate their similarity with the target domain and record performance. From the left part of Figure 5, the performance generally positively correlates with TVC. However, its precision is poor, so that cannot be used as an indicator. Points around the green line have similar TVC scores, but the performance is quite diverse, i.e., the green points' are from 20% to 70%. A similar conclusion can be drawn from the horizontal direction: blue points around the blue line have identical performance, but their TVC scores are 36% to 87%. Therefore, data selection with TVC suffers severe performance fluctuation. By comparison, there is an apparent positive linear correlation between combined similarity and performance in the target domain (See the right part of Figure 5).

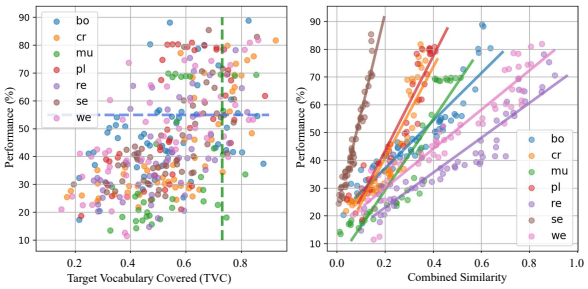


Figure 5: The relation between performance (y-axis) and the similarity function (x-axis). Different target domains are in different colors.

More analysis of (1) the comparison between the combination similarity function with its component TVC, TIS, and LO, and (2) inter-domain relations are shown in the appendix.

7 Conclusions and Future Work

In this paper, we prove the existence of negative knowledge transfer in few-shot learning and propose a similarity-based method to select pertinent data before training. We propose a Shared-Private Network (SP-Net) for the few-shot slot tagging task. We prove the effectiveness and advantages of both the data selection method and SP-Net with experiments. In the future, we will investigate the relations among domains and improve our data selection method to select episodes or samples rather than domains.

Acknowledgements

This work is supported by Innovation & Technology Commission HKSAR, under ITF Project No. PRP/054/21FX.

References

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. 2020. Few-shot text classification with distributional signatures.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations.

Xinyang Chen, Sinan Wang, Bo Fu, Mingsheng Long, and Jianmin Wang. 2019. Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces.

Haskell B Curry. 1944. The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics*, 2(3):258–261.

Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. 2019. Using similarity measures to select pretraining data for NER. 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 1460–1470, Minneapolis, Minnesota. 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.

Lin Gui, Ruifeng Xu, Qin Lu, Jiachen Du, and Yu Zhou. 2018. Negative transfer detection in transductive transfer learning. *International Journal of Machine Learning and Cybernetics*, 9(2):185–197.

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.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network.

Young-Bum Kim, Sungjin Lee, and Ruhi Sarikaya. 2017. Speaker-sensitive dual memory networks for multi-turn slot tagging. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 541–546. IEEE.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Vancouver, Canada. Association for Computational Linguistics.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020. Coach: A coarse-to-fine approach for cross-domain slot filling.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer.

Sara Meftah, Nasredine Semmar, Youssef Tamaazousti, Hassane Essafi, and Fatiha Sadat. 2021. On the hidden negative transfer in sequential transfer learning for domain adaptation from news to tweets. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 140–145, Kyiv, Ukraine. Association for Computational Linguistics.

Mansfield Merriman. 1877. *A List of Writings Relating to the Method of Least Squares: With Historical and Critical Notes*, volume 4. Academy.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

Sachin Ravi and Hugo Larochelle. 2016. Optimization as a model for few-shot learning.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. 2018. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Ruhi Sarikaya, Paul A Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiaohu Liu, et al. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *2016 ieee spoken language technology workshop (slt)*, pages 391–397. IEEE.

Darsh J Shah, Raghav Gupta, Amir A Fayazi, and Dilek Hakkani-Tur. 2019. Robust zero-shot cross-domain slot filling with example values.

Yangyang Shi, Kaisheng Yao, Hu Chen, Dong Yu, Yi-Cheng Pan, and Mei-Yuh Hwang. 2016. Recurrent support vector machines for slot tagging in spoken language understanding. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 393–399.

Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Hongru Wang, Zezhong Wang, Gabriel Pui Cheong Fung, and Kam-Fai Wong. 2021. Mcml: A novel memory-based contrastive meta-learning method for few shot slot tagging.

Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. 2019. Characterizing and avoiding negative transfer.

Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):1–37.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. TapNet: Neural network augmented with task-adaptive projection for few-shot learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7115–7123. PMLR.

Su Zhu, Ruisheng Cao, Lu Chen, and Kai Yu. 2020. Vector projection network for few-shot slot tagging in natural language understanding.

X. Zhu, D. Anguelov, and D. Ramanan. 2014. Capturing long-tail distributions of object subcategories. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 915–922.

Linguistic Knowledge in Data Augmentation for Natural Language Processing: An Example on Chinese Question Matching

Zhengxiang Wang
Department of Linguistics, Stony Brook University
zhengxiang.wang@stonybrook.edu

Abstract

To investigate the role of linguistic knowledge in data augmentation (DA) for Natural Language Processing (NLP), we designed two adapted DA programs and applied them to LCQMC (a Large-scale Chinese Question Matching Corpus) for a binary Chinese question matching classification task. The two DA programs produce augmented texts by five simple text editing operations (or DA techniques), largely irrespective of language generation rules, but one is enhanced with a pre-trained n-gram language model to fuse it with prior linguistic knowledge. We then trained four neural network models (BOW, CNN, LSTM, and GRU) and a pre-trained model (ERNIE-Gram) on the LCQMC's train sets of varying size as well as the related augmented train sets produced by the two DA programs. The results show that there are no significant performance differences between the models trained on the two types of augmented train sets, both when the five DA techniques are applied together or separately. Moreover, due to the inability of the five DA techniques to make strictly paraphrastic augmented texts, the results indicate the need of sufficient amounts of training examples for the classification models trained on them to mediate the negative impact of false matching augmented text pairs and improve performance, a limitation of random text editing perturbations used as a DA approach. Similar results were also obtained for English.

1 Introduction

Data augmentation (DA) is a common solution to the problems of limited and imbalanced data. It works by generating novel and label-preserving data from the existing data (Xie et al., 2020), which would otherwise be unavailable or expensive to collect. Owing to the increasing popularity of supervised deep learning models that demand large-scale labeled data as well as more studies on understudied/under-resourced language and text domains, the Natural Language Processing (NLP) community has seen a growing interest in DA in recent years (Feng et al., 2021; Liu et al., 2020; Shorten et al., 2021). However, unlike image and speech, whose physical features can be relatively easily manipulated without deviating from the original labels, text augmentation poses a bigger challenge. This is simply because there is no easy and automatic way to paraphrase a randomly given piece of text while preserving its linguistic integrity and, above all, meaning. As such, while there are well established and widely applied DA techniques as well as frameworks in image and speech recognition research[1] with noteworthy success (Iwana and Uchida, 2021; Park et al., 2019; Shorten and Khoshgoftaar, 2019), DA for NLP as a whole remains underexplored (Feng et al., 2021).

The main purpose of this paper is to investigate a fundamental question we found unanswered to the best of our knowledge: the role of linguistic knowledge in DA for NLP; in particular, whether more linguistic knowledge leads to a better DA approach. By a better DA approach, we mean one that can lead to superior trained models' performance on a given NLP task. Intuitively, with more linguistic knowledge instilled, a DA approach is expected to augment text of higher-quality or more grammatical and thus to be presumably better. We believe a deeper understanding of what counts as a better DA approach and the role of linguistic knowledge will trigger more in-depth experiments and discussions and advance this research area to the next stage. Eventually, these efforts will turn into potential great benefits, both academically and commercially, helping train robust NLP models with small data.

To conduct our research, we present two DA programs and train five supervised classification models on the augmented train sets for a binary Chinese question matching classification task. For simplicity and interpretability concerns, the DA programs used in this study are adapted from the Easy Data Augmentation (EDA) program (Wei and Zou, 2019), which augments text by four naïve text editing operations, largely irrespective of language generation rules. The only difference between the two adapted programs is whether they have a pre-trained statistical n-gram language model (LM) to select the most linguistically likely outputs, an effective mech-

[1] Although there is certain overlap between speech recognition and NLP, they are two independent fields with divergent concerns and specializations (Manning and Schütze, 1999). Typically, NLP is about text processing only.

anism to fuse a program with probabilistic linguistic knowledge. We choose n-gram LM over neural LMs because it is more efficient to train, and most importantly, more interpretable for its straightforward frequency-based approach. As the EDA approach has shown success (Wei and Zou, 2019) in various sentiment-related and sentence type classification tasks with small datasets (e.g., mostly around 10k examples), we choose LCQMC (a Large-scale Chinese Question Matching Corpus) compiled by Liu et al. (2018) to compare the goodness of the two adapted programs, a large labeled corpus with over 260k examples. Since our corpus is much larger and the question matching task involves comparing a pair of text, instead of one, for label prediction, it is a more reliable way to test the capacity and generalizability of a DA approach. In principle, if a DA approach can work well for the question matching task, it should also show promise for those simpler and related NLP tasks, as question matching, or text matching, is one of the most basic tasks for NLP.

The contributions of this paper are threefold. First, we present the first study on the role of linguistic knowledge in DA for NLP with a special focus on the effects of probabilistic linguistic knowledge on a DA approach or technique. Second, we propose two DA programs adapted from the EDA program. Although the adapted programs are for augmenting Chinese, several changes we made, including a new DA technique and the added n-gram LM, can be universal for tailoring the EDA program to other languages. Third, we also fill the research gaps in two understudied areas: DA for question matching classification task and DA for Chinese NLP.

The code, data, and results for this study are available at `https://github.com/jaaack-wang/linguistic-knowledge-in-DA-for-NLP`.

2 Related Works

Thus far, various DA techniques has been employed in NLP research, such as thesaurus-based (Zhang et al., 2015) and embedding-based (Wang and Yang, 2015) word replacement, random text-editing perturbation (Wei and Zou, 2019), rule-specific generation (Asai and Hajishirzi, 2020; Kang et al., 2018), back translation (Sennrich et al., 2016; Singh et al., 2019), and neural-model-based predictive text transformation (Hou et al., 2018; Kobayashi, 2018; Kurata et al., 2016) etc. Most of these studies find slight but stable performance gains for training models with augmented data for given NLP tasks, such as text classification, question answering, machine translation, for a common reason that the augmented data introduces noise to the original train set and prevents the trained models from overfitting, which improves the models' generalizability on the test set.

As the NLP community is more engaged in exploring the usefulness of DA for specific NLP tasks, we have not been able to find any focused studies from the existing literature related to the subject matter of this study, i.e., the role of linguistic knowledge in DA for NLP. However, some indirect evidence seems to be affirmative. For example, Kobayashi (2018) trained a recurrent neural network (RNN) LM, which replaces words with paradigmatic relations predicted by the RNN LM to generate new examples. Since this approach ignores the semantic association between the replaced words and the corresponding labels, he also constrained the LM to predict words more compatible with the given labels by probability. By so doing, he found about 0.2% overall improvements in accuracy for 5 sentiment-related and one question type classification tasks. According to the results reported by Kang et al. (2018), we also find that while not consistently, a sequence to sequence (seq2seq) DA model blended with a few hand-crafted rules increases more test set accuracy than the base seq2seq DA model when certain ratios of two textual inference datasets were augmented. However, since these neural DA models already encode and learn implicit linguistic knowledge through complex representation learning, it is not possible to fully recognize the effects of those added linguistic knowledge, either implicit or explicit, in them.

Relevant to our hypothesis on what counts as a better DA approach, we can find strong supports by thinking in reverse. That is, although text augmentation helps increase the size of the training texts, which then improves the performance of the trained models through regularization, it is still incomparable to the human-produced-and-annotated training texts of a same size, which by default we assume to be superior in quality as well as more diverse. For example, in Wei and Zou (2019), they augmented the original training examples by a factor of 9, giving them 5,000 training examples when 500 were given. Although the augmented train set shows average 3% performance gains in accuracy on the test set for 5 classification tasks, compared to that without augmentation, this is still significantly lower than the average 10% performance improvements when the models are trained on 5,000 of the original training examples[2]. Therefore, we expect that coupled with a n-gram LM, the adapted EDA program that utilizes random text-editing perturbations, will augment higher-quality text, and thus achieve better trained models' performance.

3 Experimental Setup

3.1 LCQMC

LCQMC contains over 260k question pairs, extracted from BaiduKnows, a Quora-like online Q&A platform. Each question pair is manually annotated by three external professional annotators with a label, 1 or 0, to represent whether a question pair matches or not in terms of the expressed intents. As judgements vary from person

[2] Wei and Zou (2019) claims that with the augmented texts, their classification models achieve higher average accuracy using only 50% of the train set than when the models are trained on the entire train set without augmentation. This is misleading since the performance of their models starts plateauing when the models see 20% of the train set.

Dataset	Total Pairs	Matched	Mismatched
Train	238,766	138,574	100,192
Dev	8,802	4,402	4,400
Test	12,500	6,250	6,250

Table 1: The basic statistics of LCQMC data sets.

to person and the interpretation of some question pairs is bound to contexts, there are about 15% annotation inconsistency and 20% annotation uncertainty (Liu et al., 2018). In this study, we keep the original separation of the train set, the development set, and the test set as is in LCQMC, whose basic statistics are shown in Table 1.

3.2 Two adapted DA programs

The base DA program developed in this study is adapted from the EDA program[3] (Wei and Zou, 2019) and the control DA program is the base program combined with a pre-trained statistical n-gram LM (refer to the next section). We name these two programs as the REDA program and the $REDA_{+NG}$ program respectively, where REDA stands for Revised Easy Data Augmentation.

Like the EDA program, the REDA program also has four text editing operations, i.e., Synonym Replacement (SR), Random Swap (RS), Random Insertion (RI), and Random Deletion (RD). Their functions are as follows: SR works by randomly replacing synonyms for eligible words based on a given dictionary, while RS works by randomly swapping word pairs. RI inserts random synonyms, if any, instead of random words, to avoid uncontrolled label change. In contrast, RD deletes words at random. We used jieba[4], a popular Chinese text segmentation tool, to tokenize Chinese text throughout this research.

To further diversify the augmented texts, we also created a new text editing operation called Random Mix (RM), which randomly selects 2-4 of the other four operations to produce novel texts. Besides, a few major changes were also made to fix few bugs we found on the EDA program and to better serve our needs of augmenting Chinese and conducting this research, including:

1. We rewrote the entire program to ensure that there are no duplicates in the augmented texts, including one for the original text. Duplicates can occur when there are no synonyms to replace (SR) or insert (RS) for words in the original texts, or when the same words are replaced or swapped back during SR and RS operations.
2. The REDA program does not preprocess the input text by removing punctuations or by introducing stop words. We did not find this type of preprocessing helpful and necessary in general or makes sense for the basic idea of random text editing behind the EDA program.
3. Instead of using WordNet for SR, we compiled a preprocessed Chinese synonym dictionary leveraging multiple reputational sources[5], including Chinese Open Wordnet[6]. Moreover, unlike the EDA program, the REDA program only replaces one word at a given position at a time, instead of replacing all its occurrences, which we see as extra edits.

The $REDA_{+NG}$ program inherits the base REDA program but additionally utilizes the n-gram LM pretrained to select the most likely augmented text(s) for each text editing operation from a variety of possible outputs. We have open-sourced two separate versions of code for these two DA programs, but during this study, we always combined them together in one working procedure so that the augmented texts outputted by these two programs are selected from the same pool. The implementation of this combination is also available at the open-sourced GitHub repository.

3.3 N-gram LM

To train the n-gram LM, we first compiled an independent corpus of BaiduKnows Q&A texts based on an existing project found on GitHub, which scrapes over 9 million question-answer pairs from BaiduKnows platform[7]. This compiled corpus contains over 654 million words (or over 1.1 billion Chinese characters). Then, the relative frequency of unigram, bigram, trigram, and 4-gram for this corpus was calculated based on words and line by line with the results saved in four separate json dictionaries as the pre-trained parameters. When counting these n-grams, we added two special tokens, <START> and <END>, in the beginning and end of each line, to keep track of their tendency to stay ahead or at the end of a line. For efficiency concerns, we adjusted the relative frequency for the unigrams simply by assigning unseen vocabulary the same frequency with those one-off unigrams and employed stupid backoff without discounting unseen non-unigrams (Brants et al., 2007). Finally, the n-gram LM takes the relative frequency of the n-grams as an estimation to their true probability of occurrence and calculates the maximum log probability of input text based on the chain rule of probability (Jurafsky and Martin, 2009) as follows:

$$\log P(NG_1 : NG_n) = \log \prod_{i=1}^{n} P(NG_i) = \sum_{i=1}^{n} \log P(NG_i)$$

where NG represents n-gram that is automatically generated by our n-gram LM. The n-gram starts with 4-gram, if any, and keeps backing off into low-order n-gram combination, if a higher-order n-gram is not available in the pre-made json dictionaries.

[3] https://github.com/jasonwei20/eda_nlp/tree/master/code.

[4] https://github.com/fxsjy/jieba.

[5] https://github.com/jaaack-wang/Chinese-Synonyms.

[6] http://compling.hss.ntu.edu.sg/cow/.

[7] https://github.com/liuhuanyong/MiningZhiDaoQACorpus.

3.4 Classification models

We chose four neural network (NN) models and one transformer-based pre-trained model as the classification models. The NN models include the Bag of Words (BOW) model, the Convolutional Neural Network (CNN) model, and two RNN models: Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). BOW model is a conventional technique to represent a text by summing up the embeddings of its words, and the similarity between texts is then often measured by Euclidean distance or cosine distance of the texts' embeddings. Since Kim (2014), CNN has been proven to be effective in various text classification tasks, including text pairing (Severyn and Moschitti, 2015). LSTM and GRU are two popular sequence models that consider word orders and have also been applied to semantic similarity tasks (Tai et al., 2015; Tien et al., 2019), which we think may be especially useful for distinguishing the augmented texts from the natural texts, and more importantly, distinguishing the casually augmented texts by the REDA program from the conditionally augmented texts by the $REDA_{+NG}$ program in terms of the test set performance. Finally, the pre-trained model ERNIE-Gram (Xiao et al., 2020) was also chosen for its state-of-the-art performance on the LCQMC dataset.

The models were constructed using Baidu's deep learning framework Paddle[8] and its NLP software PaddleNLP[9].

4 Results

4.1 Quality of the augmented texts

To evaluate the quality of the augmented texts generated by the REDA and $REDA_{+NG}$ programs, we designed three simple experiments to check their ability to restore to natural texts when modified texts or a pseudo synonym dictionary were given for three basic text editing operations, i.e., SR, RS, and RD. We skipped RI and RM because inserting random synonyms is generally not the natural way of language use however (un)natural the input text is and the text quality resulting from RM can be inferred from the other basic operations directly.

The experiments went as follows. For SR, we designed a pseudo synonym dictionary made up of 3855 one-word-four-synonym pairs, where every word is mapped to four pseudo synonyms, one being the word itself and the rest non-synonym random words. All the words in the dictionary are those whose frequencies rank between the 1000th and the 10000th place in the unigram dictionary complied for the n-gram LM. For RS and RD, we randomly reordered the natural texts and added random words sampled from the texts respectively before RS and RD were performed. 10,000 pieces of texts were randomly sampled from the LCQMC's train set for 5 times for every comparison we made. The average accuracy scores are reported in Table 2.

		One Edit	Two Edits	Three Edits
SR	REDA	22%	6%	2%
	+N-gram	**88%**	79%	64%
RS	REDA	9%	4%	4%
	+N-gram	**69%**	41%	34%
RD	REDA	16%	5%	2%
	+N-gram	**39%**	22%	15%

Table 2: The average accuracy scores of the two DA programs in three text restoration tasks based on different number of edits. SR: Synonym Replacement; RS: Random Swap; RD: Random Deletion. Best performance given a DA technique is highlighted in **bold**.

As can be seen, while both programs' performance declines as the number of edits increase, the $REDA_{+NG}$ program always outperform the REDA program in restoring to the natural texts. In fact, for the REDA program, restoring the modified texts to the original ones is a matter of chance equal to the inverse of the number of possible outputs available. However, the $REDA_{+NG}$ program augments texts of maximum likelihood, which tends to be closer to the natural texts expected. This is also true when natural texts are given as inputs. For example, through manual inspections, we found the $REDA_{+NG}$ program does much better in selecting the appropriate synonyms according to the linguistic contexts, which is a problem for the REDA program due to the ubiquitous existence of polysemy. By measuring the bigram overlap rate and edit distances of output texts randomly swapped twice from the natural texts, we found that the average overlap rate for the REDA program is much lower (i.e., 0.29 versus 0.77) and the average edit distances are much larger (i.e., 3.0 versus 1.4) than the $REDA_{+NG}$ program, meaning the latter preserves more collocational features of the natural texts and thus augments higher-quality texts.

Nevertheless, the $REDA_{+NG}$ program is also not free of considerable text quality decrease when more text edits are performed. This is largely due to the drastic increase of possible output texts as well as the more likely semantic shift of the original texts with large proportion of the input texts changed. Therefore, to conduct our research, the number of text edits performed is set proportional to the number of words of the input texts, so that a large quality difference of the augmented texts by the two programs can be maintained. More concretely, in the study, we set the SR and SR rate at 0.2 and the RI and the RD rate at 0.1 and applied Python rounding rules[10]. RM will only randomly select two of the other four text editing operations with one text edit each for every input text to make the study more controlled.

[8] https://github.com/PaddlePaddle/Paddle

[9] https://github.com/PaddlePaddle/PaddleNLP

[10] When an even number ends with ".5", it will be rounded down; otherwise, rounded up.

LCQMC	5,000	10,000	50,000	100,000	238,766
REDA	66,267	132,513	563,228	929,176	2,218,512
+N-gram	64,358	128,649	544,583	893,779	2,133,163

Table 3: The train set size for the corresponding REDA and $REDA_{+NG}$ augmented train sets.

Models	5k	10k	50k	100k	Full Set	Average
BOW	59.4%	60.4%	65.4%	67.8%	73.8%	65.4%
+REDA	58.1%	60.9%	68.2%	72.2%	76.4%	67.2%
+$REDA_{+NG}$	58.8%	59.6%	68.1%	71.2%	76.0%	66.7%
CNN	59.3%	63.4%	67.2%	69.0%	72.9%	66.4%
+REDA	59.8%	62.6%	66.8%	69.8%	74.9%	66.8%
+$REDA_{+NG}$	60.3%	62.0%	67.9%	69.1%	74.0%	66.7%
LSTM	60.0%	62.1%	66.2%	69.6%	74.8%	66.5%
+REDA	58.9%	61.5%	67.7%	71.8%	76.4%	67.3%
+$REDA_{+NG}$	57.7%	60.9%	67.7%	71.7%	75.9%	66.8%
GRU	59.8%	61.9%	68.1%	70.3%	76.8%	67.4%
+REDA	58.7%	61.3%	68.7%	72.7%	76.8%	67.6%
+$REDA_{+NG}$	58.8%	60.0%	67.8%	72.5%	76.6%	67.1%
ERINE-Gram	**78.7%**	**81.7%**	**85.9%**	**87.1%**	**87.4%**	**84.2%**
+REDA	77.5%	80.3%	84.1%	85.0%	85.7%	82.5%
+$REDA_{+NG}$	78.6%	80.1%	83.8%	84.6%	85.8%	82.6%
Average	63.5%	65.9%	70.6%	72.8%	77.1%	70.0%
+REDA	62.6%	65.3%	71.1%	74.3%	78.0%	70.3%
+$REDA_{+NG}$	62.8%	64.5%	71.1%	73.8%	77.7%	70.0%

Table 4: Test set accuracy of the five classification models trained on the three types of train sets of varying size. Best performance given a train set size (of original training examples) is highlighted in **bold**.

4.2 Effects of the two DA programs

We trained the five classification models in Baidu Machine Learning (BML) CodeLab on its AI Studio[11] with Tesla V100 GPU and 32GB RAM. The models were trained with 64 mini batches, a fixed 5e-4 learning rate (5e-5 for ERNIE-Gram model), and constantly 3 epochs. We used Adaptive Moment Estimation (Adam) optimizer and cross entropy loss function. We kept the original development set for validation purposes.

The following training sizes were experimented: 5k, 10k, 50k, 100k, and full size, approximately equal to 2%, 4%, 21%, 42%, and 100% of the LCQMC's train set respectively. When the train set size is 5k and 10k, we augmented two new texts for SR and RS, and one new text for RI, RD, and RM, because the last three text editing operations show smaller differences for the REDA and $REDA_{+NG}$ programs in terms of text quality (refer to the last section), which we want to hold as large as possible for the sake of this research. That translates into maximum 7 new texts for every text and up to 14 new texts for every text pair due to deduplication. Every augmented text was crossed paired with the other text that was a pair to the text being augmented with the original label kept for the newly made text pair. To make the training more manageable, we only augmented 5 new texts for every text with one output for every text editing operation, meaning a maximum tenfold increase in size when the associated train set size is 50k and more. The corresponding augmented train set size is given in Table 3.

[11] https://aistudio.baidu.com/aistudio/index

The accuracy scores as well as the average precision, recall, and F1 scores on the test set are presented in Table 4 and Table 5, respectively. Contrary to our expectation, we do not find that the $REDA_{+NG}$ augmented train sets lead to better test set performance than the REDA augmented train sets, when it comes to the four metrics used in this study. According to the pairwise Mann-Whitney U tests we ran, there is no statistically significant difference across the four metrics among each type of models trained on the two types of augmented train sets, as the p-values were constantly far greater than .05. Although the former program does produce higher-quality augmented texts from a linguistic perspective as discussed above, evidence shows that models trained on the REDA augmented train sets outperform those trained on the $REDA_{+NG}$ augmented train sets by an average 0.3% both in the accuracy and F1 scores. As can be seen from Table 4, the $REDA_{+NG}$-led models only outperform the REDA-led ones in terms of the test set accuracy when the train set size is 5k for four models except the LSTM model and when the ERNIE-Gram models were fine-tuned on the full augmented train sets. Moreover, for any classification model trained on the REDA augmented train sets, in most cases, it achieves a slightly better score for the four metrics than the model trained on the $REDA_{+NG}$ augmented counterparts. It follows that the role of probabilistic linguistic knowledge instilled in the $REDA_{+NG}$ program is overall minimal and sometimes harmful to DA applied to the binary question matching task.

Also noticeable from Table 4 is that 50k training ex-

Models	Baseline			REDA			$REDA_{+NG}$		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BOW	61.5%	82.5%	70.4%	63.3%	81.7%	71.3%	62.9%	81.8%	71.1%
CNN	62.8%	80.5%	70.5%	63.6%	78.1%	70.0%	63.8%	76.2%	69.3%
LSTM	62.5%	82.7%	71.2%	63.4%	81.4%	71.3%	63.0%	82.1%	71.3%
GRU	63.4%	82.4%	71.7%	63.8%	81.9%	71.7%	63.3%	81.7%	71.4%
ERINE-Gram	**78.0%**	95.8%	**85.9%**	75.8%	**95.9%**	84.6%	76.0%	95.3%	84.6%
Average	65.6%	84.8%	73.9%	66.0%	83.8%	73.8%	65.8%	83.4%	73.5%

Table 5: Average test set precision, recall, and F1 scores for the five classification models trained on the three types of train sets. Best performance given a metric (precision, recall, or F1) is highlighted in **bold**.

amples appear to be the threshold where the two DA programs start bringing gains to the related test set accuracy scores compared to the baselines, except for the finetuned ERNIE-Gram models. However, as shown in Table 5, there is also a gap in the recall scores in favor of the baseline models, which may be attributed to the false matching text pairs produced by the two DA programs due to the inability of the underlying text editing operations to make strictly paraphrastic augmented texts. But these noisy augmented texts in return enable the classification models to generalize better on those matching text pairs judged to be non-matching by the baseline models, as indicated by the average larger precision scores. In addition, the advantage of the pre-trained model over the traditional NN models is significant: the ERNIE-Gram models, finetuned on all the three types of train sets, show about 12% to 17% average gains across the four metrics in relation to the other four trained models. This shows the promise of applying transfer learning to DA for NLP, which may be worth further studying in the future.

4.3 Ablation study: each DA technique

To gain a more nuanced understanding of the role of linguistic knowledge in each one of the DA techniques performed by the two DA programs, we conducted an ablation study where we trained models on train sets augmented by only one DA technique. That means, for a train set of given size randomly sampled from the LCQMC's train set, there are five types of corresponding augmented train sets. Our analyses are based on comparing the average test set performance of the five models trained on the three types of train sets for the five augmentation scenarios. We also excluded ERNIE-Gram models, which are revealed to be distinct from the rest models across the four metrics in the last section, to see if there is a noticeable difference.

As the training sizes are shown to have an effect on whether the DA-led models outperform the baseline models, to further validate that, we chose 11 training sizes for this ablation study, namely, 5k, 10k, 25k, 50k, 75k, 100k, 125k, 150k, 175k, 200k, and full set, roughly equal to 2%, 4%, 10%, 21%, 31%, 42%, 52%, 63%, 73%, 84%, and 100% of the LCQMC's train set respectively. The basic hyperparameters are same with the previous section. However, to make the training more manageable, we only trained 2 epochs when the baseline training size is 50k or 100k and 1 epoch when the baseline training size is over 100k for the three types of train sets. Since it is evident from Table 4 that a larger training size under the same condition always leads to a higher test set performance, spending extra time in training a total of 605 models[12] with fixed 3 epochs may thus not be worthwhile to re-verify. Moreover, we only augmented 2 texts per text per DA technique when the baseline training size is no less than 50k and 1 text when otherwise, with the cross pairing applied, similar to what we did in the previous section. Please refer to the Appendix for more details.

Figure 1 shows the average test set accuracy scores of the five classification models trained on the three types of train sets under different text editing conditions and across different training sizes. In line with the previous finding, the effect of probabilistic linguistic knowledge on each one of the five DA techniques is minimal and of no statistically significant difference, both individually and on average. Although with certain text editing operations, such as RS, RI, and RM, there exist several points in which there is a relatively large difference in the accuracy scores between the two DA-led models, these differences fluctuate along the x-axis and eventually get reduced to be negligible when the average performance are concerned. This basic pattern remains true when we plotted the average test set performance based on any one of the four metrics with or without the ERNIE-Gram models.

Also related to the previous finding is that there does exist a threshold where the DA-led models outperform the baseline models in the test set accuracy scores, which appears to be the 100k training size or so, instead of 50k as in Table 4. The discrepancy may be explained by the different epoch numbers (e.g., 2 vs 3 for 50k) and possibly more importantly the separation of the DA techniques, which, however, are beyond the scope of this study. We also examined plots based on the other three metrics with or without the ERNIE-Gram models to explore the cause of such phenomenon. Figures 2 and 3 present the average test set precision and

[12] Since there are 11 training sizes and 5 classification models, that translates into 55 models for the baseline train sets. As there are 5 DA techniques applied in 2 different ways (with or without n-gram LM), that translates into 550 (55 * 5 * 2) models for the augmented train sets. Hence, we have 605 models to train in total.

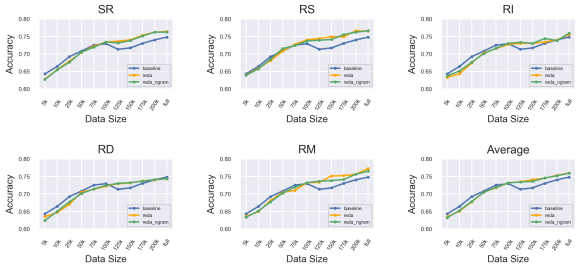


Figure 1: Average test set accuracy scores of the three models under different conditions (i.e., text editing type, training data size) for the two types of LCQMC's train sets. The sixth plot averages the statistics of the previous five plots.

recall scores of the five classification models trained on the three types of train sets respectively. As can be seen, there is no general trend in which the baseline models surpass the DA-led counterparts in the test set recall scores, but a similar pattern that resembles that of Figure 1 also exists in Figure 2. That means, the increase in the precision scores, after certain amounts of training examples are trained, are the main driver that makes the baseline models outperformed by the DA-led ones in terms of test set accuracy scores as well as the F1 scores, which are not shown here to save space. Moreover, this conclusion also largely holds when the ERNIE-Gram models are excluded.

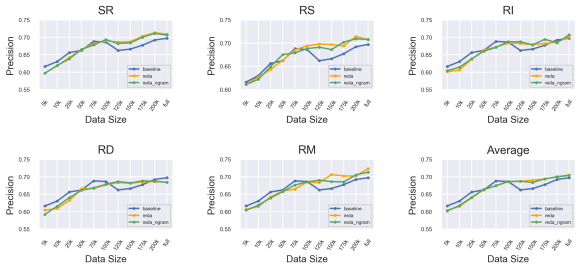


Figure 2: Average test set precision scores of the five classification models.

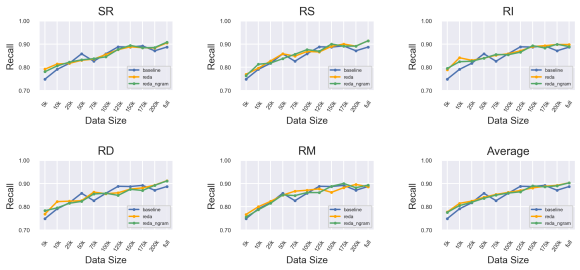


Figure 3: Average test set recall scores of the five classification models.

5 Discussions and Conclusions

In this study, we examined the effects of linguistic knowledge on DA for a binary Chinese question matching task. We proposed two DA programs, i.e., the REDA and $REDA_{+NG}$ programs, that augment text by five random text editing operations (or DA techniques), with the $REDA_{+NG}$ program combined with a n-gram LM to fuse it with probabilistic linguistic knowledge. Surprisingly, we found that the $REDA_{+NG}$-led classification models did not surpass the REDA-led counterparts in the test set performance (i.e., accuracy, precision, recall, and F1 scores), which is also true when the five DA techniques in the two programs are applied and compared separately. In other words, our study indicates strongly that instilling more linguistic knowledge into a DA approach or technique does not necessarily make it a better one when it comes to training a better question matching classifier for Chinese, although doing so may make the augmented texts higher quality from a pure linguistic point of view.

However, since the two DA-led models achieve very close scores in the four metrics with trivial advantages for the REDA-led models, it is not possible for us to explain why adding probabilistic linguistic knowledge as a constrain does not make a meaningful difference, positive or negative. A possible explanation might be that as the five deep learning models compare a pair of texts in vector space and the way how word embeddings encode linguistic knowledge is different from humans, performing simple text editing operations in two different ways (i.e., random, conditional) on a text may result in different meanings for humans, but that for machines nevertheless is less distinguishable in the high dimension of vector space. Moreover, as we only used probabilistic linguistic knowledge as a filter to select augmented texts closer to human language use, the inherent inability of the underlying text editing operations made by the two DA programs to produce strictly paraphrastic augmented texts means the two types of augmented texts are to a considerable extent comparable in that they are mostly not the paraphrases to the original texts being augmented. However, such interpretation cannot explain why the REDA-led models often outperform the $REDA_{+NG}$-led ones by a slight but consistent margin.

Unlike Wei and Zou (2019) who show general success of their EDA program in bring performance gains for several sentiment-related and text type classification tasks across train sets of varying sizes, we only found such gains when the classification models were trained with sufficient amounts of training examples. As we expected in the beginning, question matching presents a more difficult and fundamental classification task because it involves comparing a pair of texts, instead of a single text, to predict the label for the given text pair. This nature makes question matching, or text matching in general, inherently much more sensitive to and subject to some tiny semantic changes caused by text augmentation. To further validate this hypothesis, we adjusted the two REDA programs and ran a post hoc experiment similar to Section 4.2 for English using the

Models	10k	50k	100k	150k	Full Set (260k)	Average
BOW	64.4%	69.9%	72.1%	74.2%	77.7%	71.7%
+REDA	62.5%	68.5%	71.6%	74.8%	78.0%	71.1%
+$REDA_{+NG}$	62.9%	69.4%	74.0%	75.5%	78.2%	72.0%
CNN	66.1%	71.1%	72.6%	73.4%	75.9%	71.8%
+REDA	63.7%	69.9%	72.7%	75.3%	77.6%	71.8%
+$REDA_{+NG}$	63.5%	69.3%	72.7%	74.7%	77.7%	71.6%
LSTM	65.7%	**71.6**%	72.9%	75.0%	77.9%	72.6%
+REDA	64.0%	69.8%	72.5%	75.1%	78.1%	71.9%
+$REDA_{+NG}$	64.9%	70.3%	72.7%	75.0%	78.1%	72.2%
GRU	**67.2**%	71.0%	**74.3**%	74.7%	77.4%	72.9%
+REDA	63.3%	70.0%	72.8%	74.8%	78.1%	71.8%
+$REDA_{+NG}$	64.0%	70.2%	73.8%	**75.7**%	**78.9**%	72.5%
Average	65.9%	70.9%	73.0%	74.3%	77.2%	72.3%
+REDA	63.4%	69.6%	72.4%	75.0%	78.0%	71.7%
+$REDA_{+NG}$	63.8%	69.8%	73.3%	75.2%	78.2%	72.1%

Table 6: Test set accuracy of four classification models trained on the three types of train sets of QQQD with varying sizes. Due to cost concerns, we did not finetune a pre-trained model, such as BERT, this time.

Models	Baseline			REDA			$REDA_{+NG}$		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BOW	70.9%	73.5%	72.1%	69.2%	76.1%	72.5%	71.1%	74.4%	72.7%
CNN	70.5%	75.4%	72.8%	70.7%	76.0%	73.1%	70.2%	76.5%	73.1%
LSTM	70.5%	**78.2**%	**74.1**%	70.5%	75.4%	72.8%	71.4%	74.1%	72.7%
GRU	**71.8**%	75.5%	73.5%	69.8%	76.9%	73.2%	71.6%	74.5%	73.0%
Average	70.9%	75.6%	73.1%	70.1%	76.1%	72.9%	71.1%	74.9%	73.9%

Table 7: Average test set precision, recall, and F1 scores for the four classification models trained on the three types of train sets of QQQD.

Quora Question Pairs Dataset (QQQD)[13], from which we created three label-balanced data sets of comparable sizes to the LCQMC counterparts. The average test set accuracy scores in Table 6 clearly show that models trained on the augmented train sets also need to see ample original training examples (near 150k or above) to stably outperform the baseline models, although the threshold is higher here. Therefore, for random text editing DA approach to work for question matching, there is a need of sufficient training examples to enable the trained models to mediate the negative impact of the false matching augmented text pairs resulting from random text editing perturbations and turn it into a means of regularization that improves the models' generalizability. This is a general limitation of random text editing perturbations applied as a DA approach.

Lastly, comparing the results from these two experiments, or between Table 4 and Table 6, and between Table 5 and Table 7, we can see that the discussions and conclusions drawn from the LCQMC experiment mostly apply for the QQQD experiment as well, since the obtained data shares similar patterns. Besides the threshold difference noted above, which may be dataset specific, a noteworthy difference is that $REDA_{+NG}$-led models slightly but consistently outperformed the REDA-led counterparts of test set accuracy and precision, although there is also no statistically significant difference and the average F1 scores are same. This fact again demonstrates the difficulty of fully accounting for modern deep learning experiments, but it also strongly confirms the negligible role of probabilistic linguistic knowledge in text augmentation.

[13] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

6 Limitations and future studies

Although we are highly confident that observations made in this study are reliable, we were nevertheless unable to experiment with different initializations of the two REDA programs and different configurations of the classification models, constrained by available resources. Moreover, systematically and fairly evaluating a DA approach for NLP is uneasy or even unknown. The current study only illustrates a tip of the iceberg.

In light of the limitations above, future studies may carry out similar experiments with differing setups, different NLP tasks, or even distinct methods of fusing a DA approach or technique with linguistic knowledge. Because of the simplicity and low cost of the five DA techniques employed in this study, it may also be important to re-examine the effectiveness and limitations of these random text editing operations for assorted NLP tasks. This may then give us some useful insights into building cheap and (highly) universal DA techniques for NLP, which is currently lacking in the field.

References

Akari Asai and Hannaneh Hajishirzi. 2020. Logic-guided data augmentation and regularization for con-

sistent question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5642–5650, Online. Association for Computational Linguistics.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic. 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.

Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. 2018. Sequence-to-sequence data augmentation for dialogue language understanding. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1234–1245, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Brian Kenji Iwana and Seiichi Uchida. 2021. An empirical survey of data augmentation for time series classification with neural networks. *PLOS ONE*, 16(7):1–32.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2st edition. Prentice Hall PTR, USA.

Dongyeop Kang, Tushar Khot, Ashish Sabharwal, and Eduard Hovy. 2018. AdvEntuRe: Adversarial training for textual entailment with knowledge-guided examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2418–2428, Melbourne, Australia. Association for Computational Linguistics.

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

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. 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 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Labeled Data Generation with Encoder-Decoder LSTM for Semantic Slot Filling. In *Proc. Interspeech 2016*, pages 725–729.

Pei Liu, Xuemin Wang, Chao Xiang, and Weiye Meng. 2020. A survey of text data augmentation. In *2020 International Conference on Computer Communication and Network Security (CCNS)*, pages 191–195.

Xin Liu, Qingcai Chen, Chong Deng, Huajun Zeng, Jing Chen, Dongfang Li, and Buzhou Tang. 2018. LCQMC:a large-scale Chinese question matching corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1952–1962, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In *Proc. Interspeech 2019*, pages 2613–2617.

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.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 373–382, New York, NY, USA. Association for Computing Machinery.

Connor Shorten and Taghi M. Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:1–48.

Connor Shorten, Taghi M. Khoshgoftaar, and Borko Furht. 2021. Text data augmentation for deep learning. *Journal of Big Data*, 8:1–34.

Jasdeep Singh, Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2019. Xlda: Cross-lingual data augmentation for natural language inference and question answering.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Nguyen Huy Tien, Nguyen Minh Le, Yamasaki Tomohiro, and Izuha Tatsuya. 2019. Sentence modeling via multiple word embeddings and multi-level comparison for semantic textual similarity. *Information Processing & Management*, 56(6):102090.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, Lisbon, Portugal. Association for Computational Linguistics.

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 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Dongling Xiao, Yu-Kun Li, Han Zhang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie-gram: Pre-training with explicitly n-gram masked language modeling for natural language understanding. arXiv.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Appendix

A. Size of augmented train sets for the ablation experiment

Table 8 contains the size of the train sets for the ablation experiment on LCQMC. Please note that, for simplicity, 240k is used to refer to the full size of LCQMC, which is 238,766 to be exact. Also, due to deduplication, different text editing operations may result in augmented train sets with non-trivial difference in size, as discernible in Table 8.

B. Average test set performance for the ablation experiment

Figure 4, 5, and 6 show the average test set performance (accuracy, precision, and recall, respectively) of the four classification models, excluding the ERNIE-Gram model. It is clear that the observations made in the section 4.3 still hold.

C. Size of QQQD-related data sets

We created three label-balanced data sets based on QQQD's original train set since the test set is made unlabeled for online competition. The size of the created train, development, and test sets is 260k, 20k, and 18,526, respectively. Table 9 shows the size of augmented train sets for QQQD.

Size	SR	RS	RI	RD	RM
5k	24,402	24,758	16,733	16,780	24,859
10k	48,807	49,575	33,090	33,208	49,652
25k	122,358	124,040	83,329	83,592	124,237
50k	244,577	248,074	166,839	167,296	248,539
75k	220,843	223,497	162,563	162,972	224,026
100k	294,516	297,987	216,540	217,012	298,620
125k	368,078	372,536	270,957	271,552	373,266
150k	441,643	446,941	325,027	325,738	447,838
175k	515,229	521,484	379,352	380,214	522,535
200k	588,901	595,977	433,521	434,469	597,084
240k	703,077	711,631	517,492	518,664	712,852

Table 8: Size of the augmented train sets for the ablation experiment on LCQMC.

Size	REDA	+N-gram
10k	148,341	141,604
50k	543,066	512,176
100k	1,086,063	1,023,777
150k	1,629,178	1,536,285
260k	2,823,733	2,662,639

Table 9: Size of the augmented train sets for QQQD.

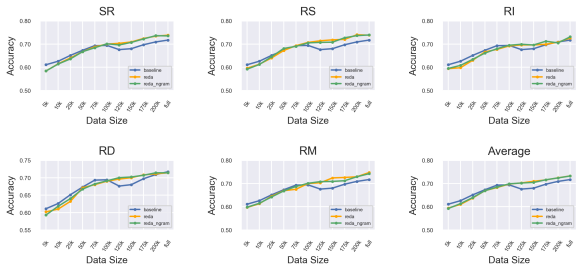


Figure 4: Average test set accuracy scores of the four classification models excluding ERNIE-Gram.

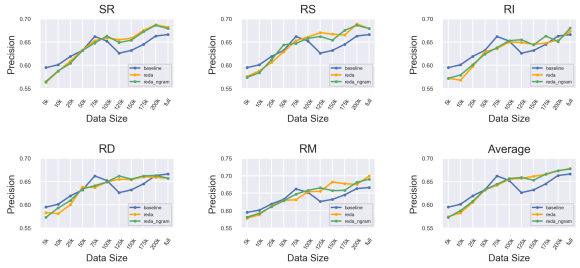


Figure 5: Average test set precision scores of the four classification models excluding ERNIE-Gram.

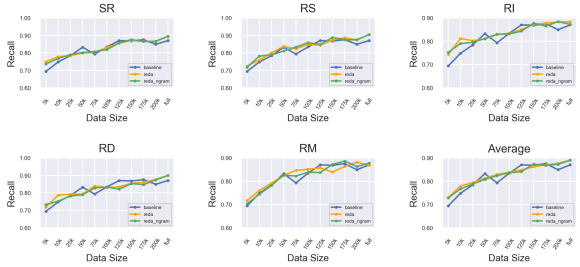


Figure 6: Average test set precision scores of the four classification models excluding ERNIE-Gram.

Detecting Security Patches in Java Projects Using NLP Technology

Andrea Stefanoni[1,2,3], **Šarūnas Girdzijauskas**[2], **Christina Jenkins**[3], **Zekarias T. Kefato**[2], **Licia Sbattella**[1], **Vincenzo Scotti**[1], and **Emil Wåreus**[4]
[1]DEIB, Politecnico di Milano, Via Golgi 42, 20133, Milano (MI), Italy
[2]KTH Royal Institute of Technology, Brinellvägen 8, 114 28 Stockholm, Sweden
[3]Devoteam Creative Tech, Klara Östra Kyrkogata 2B, 111 52 Stockholm, Sweden
[4]Debrided AB , Nordenskiöldsgatan 17, 211 19 Malmö, Sweden
andrea2.stefanoni@mail.polimi.it sarunasg@kth.se
christina.jenkins@devoteam.com zekarias@kth.se
licia.sbattella@polimi.it vincenzo.scotti@polimi.it
emil.wareus@microfocus.com

Abstract

Known vulnerabilities in software are solved through security patches; thus, applying such patches as soon as they are released is crucial to protect from cyber-attacks. The diffusion of open source software allowed to inspect the patches to understand whether they are security related or not. In this paper, we propose some solutions based on state-of-the-art deep learning technologies for Natural Language Processing for security patches detection. In the experiments, we benchmarked our solutions on two data sets for Java security patches detection. Our models showed promising results, outperforming all the others we used for comparison. Interestingly, we achieved better results training the classifiers from scratch than fine tuning existing models.

1 Introduction

The use of *Open Source Software* (OSS) has become a common practice in proprietary projects, especially thanks to the speed up in software production and the costs reduction (Vaughan-Nichols, 2015). However, this practice comes with the risk of introducing vulnerabilities in private code-bases. In this context, we introduce the concept of *security patches*: a security patch is a special type of code patch, which is a set of changes to be applied to some software to update, fix, or improve it. These security patches are designed to solve code vulnerabilities that causes the exposure to cyber-attacks.

The aforementioned code vulnerabilities have been categorised using different notations. The most famous are the *Common Vulnerabilities and Exposures* (CVE) and the *Common Weakness Enumeration* (CWE); both provide a description of the vulnerabilities discovered and the second one is organised hierarchically. Moreover, there exists data bases containing a list of vulnerable commits (changes to the software code base) like the *National Vulnerability Database* (NVD) and the *Software Assurance Reference Dataset* (SARD), which offer an helpful reference to understand these vulnerabilities. They contain examples of vulnerable code paired with the non-vulnerable counterpart, thus providing test cases for software production.

In this work we focus on OSS projects in Java maintained on *GitHub*. On this platform, a commit represent an update to the code base and is composed of two parts: *commit message* (a short description in natural language of the updated piece(s) of code) and *patch* (sometimes called *code changes*, it consists of one or more *hunks*). Hunks are the differences between the old version and the new version of source code files. These hunks are usually surrounded by context lines of the original untouched source code and marked with line numbers. Deleted rows are marked with an initial −, while the added rows start with a +.

Usually, in the process of software development, the software maintainers are overwhelmed by the number of patches released in their dependencies, which can refer to one of those OSS projects. Since applying patches requires extra work and downtime, it is important to prioritise security patches. In this sense, we propose a method based on Natural Language Processing (NLP) technologies to analyse the code modified in the patch, focusing on the semantics expressed in the code, to detect security patches, and thus allow to prioritise them.

We organise this paper as follows: in Section 2 we present the related research works for code analysis and classification, in Section 3 we presents the data sets we used as benchmarks in the experiments, in Section 4 we provide an overview of the models we considered and how we used them to tackle the detection task, in Section 5 we present the experimental approach we followed and the re-

sults we obtained, and in Section 6 we provide final remarks and propose possible future extensions.

2 Related work

NLP is the area of *Artificial Intelligence* (AI) focused on the analysis and synthesis of human language. Recently, the introduction of *Deep Learning*-based techniques in this area has pushed significantly forward the state-of-the-art on many problems. In particular, the development of *Deep Probabilistic Language Models* based on the *Transformer Architecture* (Vaswani et al., 2017) like *GPT* (Brown et al., 2020), *BERT* (Devlin et al., 2019) and *T5* (Raffel et al., 2020) seems to have enabled an impressive step forward. These models for sequence analysis are pre-trained on large text data sets doing simple tasks like next token/word prediction and can be fine-tuned for any problem, yielding impressive results due to the informative hidden representations learnt during pre-training.

These same models and techniques used for natural language, can be also applied for artificial languages, such as programming languages. In fact, according to the *Naturalness Hypothesis* (Hindle et al., 2016; Allamanis et al., 2018), we can treat source code in the same way of a document written in plain natural language. As a result, deep learning models for sequence and graph processing have been actively used to process code, including vulnerability classification (Otter et al., 2018; Semasaba et al., 2020; Wu, 2021).

The application of deep learning techniques to source code analysis evolved similarly to natural language. Early solution tackled the problem of extracting a distributed continuous representation of code pieces similarly to early works for NLP based on embeddings (i.e., vector semantic representations).

Initially, *word embedding* models for NLP used static and shallow embedding matrices to project words into compact and dense representations (Mikolov et al., 2013a,b; Pennington et al., 2014; Bojanowski et al., 2017). Such word representations can be further combined to obtain semantic vectors representing sentences (Pagliardini et al., 2018; Arora et al., 2017; Zhelezniak et al., 2019; Muffo et al., 2021, 2022) or even entire documents (Le and Mikolov, 2014; Chen, 2017; Hosseini et al., 2022).

Following these approaches, *Code2Vec* (Alon et al., 2019) was developed to extract distributed representations of the tokens in a piece of code. However, Code2Vec exploits more complex structures than vanilla word emebedding models, like *Abstract Syntax Trees* (AST), to compute the vector representations.

More recently, models for contextual representation from sequence analysis have emerged: *CodeBERT* (Feng et al., 2020) , for instance, employs the BERT auto-encoder to carry out source code and natural language analysis, serving as impressive feature extraction model that can be used on many downstream tasks, including vulnerability detection. There are also pre-trained models trained directly for patch analysis, like *CommitBERT* (Jung, 2021), however their accessibility is still limited.

Besides feature extraction for code analysis, many works focused also on specific tasks. In the context of security patch detection/classification, many solution work on C/C++ data sets (due to to higher data availability) and employ multiple sub-models to break down the input analysis.

In the case of *SPI* (Zhou et al., 2022) and *PatchRNN* (Wang et al., 2021), both models use multiple *Long Short-Term Memory* (LSTM) (Hochreiter and Schmidhuber, 1997) networks fed using Word2Vec embeddings trained on C code tokens. SPI uses two LSTMs to extract features respectively from the added and deleted lines of code in a patch and it further enhance the input with the commit message to carry out the classification. PatchRNN uses a twin LSTM solution to analyse the code before and after the patch with the information from the commit message to classify the patch. Both models encode the commit message using standard embedding techniques, namely Word2Vec, and use a mixture of experts to combine the results of code analysis with that of the commit messages. Differently, *CC2Vec* (Hoang et al., 2020) processes only the code changes and exploits the hierarchical structure of a patch (divided into *token*, *line*, and *hunk*) thorough a *Hierarchical Attention Network* (HAN). It analyses with two separate networks added and deleted lines and the post-process together the extracted feature vectors. This last approach was employed also for the classification of C language patches to identify the stable ones.

Concerning Java-specific solutions for security patches classification, *Commit2Vec* (Lozoya et al., 2021) represent the closest work to the one we apresent in this paper. However, the data set used by Commit2Vec is only partially available, making

impossible a direct comparison with our work. The Commit2Vec model is based on Code2Vec embeddings: it encodes the AST of previous and current versions code (with respect to the patch), then an attention layer further processes the embedded code differences to perform the classification.

All the aforementioned works use binary classes division to categorise the security patches, while we are also interested in macro-classes identification.

Recent results showed that handcrafted features and a *Random Forest* classifier (Breiman, 2001) are sufficient to obtain reasonable performances on a set of ten macro-classes derived by the original CVE labels (Wang et al., 2020).

3 Data

To the end of this work, we focused only on Java security patches. In particular, we used three separate data sets: the first two are private data collections, while the latter is publicly available and was curated by Ponta et al. (2019).

We merged the first two data sets into one comprising 123 000 samples (i.e., code patches, with 1157 being related to security issues). Samples from the former data set use a binary labelling system, while those from the latter used both CVE and CWE notations. After merging, labels were uniformed to the binary system with the two classes being *security* and *non-security*. The training set was composed of 933 and 918 samples (respectively for the two classes) and the test set was composed of 224 and 239 samples (respectively for the two classes). To cope with the unbalance in the data set we undersampled the non-security class.

The third data set (Ponta et al., 2019) is composed of 1175 security patches labelled with the CVE notation. Due to the high number of different classes, that would have prevented effectively training a classifier, we first converted the CVE notation to CWE (yielding 605 different classes instances) and then we clustered manually the resulting labels down to five:

Improper management of resources patches to solve vulnerabilities connected to resources and variables (e.g., buffer overflow).

Cryptography features patches to solve vulnerabilities connected to data security and information leakage.

Authentication errors patches to solve vulnerabilities connected to access control, authentication, and user sessions.

Other all the security patches that don't fall under the previous categories (e.g., channel errors).

Non-security complementary class to the security patches (e.g., bug fixes, new features, etc.). Samples from this class were taken randomly from the first two data sets.

Pre-processing steps of all data sets consisted in:

- the extraction of added and deleted lines from the patches;
- replacement of comments, strings, and numbers with as many special tokens;
- splitting of function and variable names (we used the most common naming conventions like *snake case*, *camel case*, and *kebab case*);
- deletion of special characters and stopwords (with the exception of java specific ones).

We divided code tokens on spaces and lowercased to all non-special tokens.

4 Methodology

In the following, we describe how we encoded the input sequence representing the code to analyse and the neural network models we considered to carry out the classification task. We distinguished between baseline models, used to get an idea of the performances achievable on the considered data sets, and advanced models, which exploit more complex architectures to obtain the best results.

4.1 Embedding

As happens for natural language, we converted the sequence of tokens written in Java into a continuous vector representation compatible with neural networks. For this task we considered different embedding models:

Uninitialised embeddings we employed 32-dimensional randomly initialised embeddings we trained with the overall models.

Word2Vec we trained a 100-dimensional embedding model on the code contained in the private data sets.

Code2Vec we resorted to a pre-trained model with 128-dimensional embeddings.

Tests showed that uninitialised embeddings yield a better representations for our task. This is also supported by the results we report in Section 5: uninitialised models achieve the best scores.

4.2 Baseline models

We considered two baseline classification models:

XGBoost (Chen and Guestrin, 2016) we trained this model on handcrafted features, similar to those used by Wang et al. (2020), and we employed a count encoder for the patch.

LSTM we employed this baseline similarly to the work on Commit2Vec, we employed this baseline; however, we fed it with the added and deleted lines concatenated with a special separator token.

4.3 Advanced models

As premised, a part from the baselines, we considered more complex models. For many of them we considered a base version and the *patch* version, were the internal model is replicated to analyse separately added and deleted lines as in the work on PatchRNN. We leveraged both pre-trained models coming from previous works or generic uninitialised models:

PatchRNN inspired by the original work, we used twin recurrent networks to encode separately added and deleted lines. We used *Gated Recurrent Units* (GRU) (Cho et al., 2014) with 64 hidden units to build this model.

HAN as for the PatchRNN, we took inspiration from the HAN used in CC2Vec, and implemented a three layer version of it (respectively for word, hunk and file level). In each layer we used GRUs, with 64 hidden units, and attention was computed on top of it. During the hyperparameters search, we fixed the number of files to two and hunks to three for the sake of parallelisation.

CodeBERT we employed a pre-trained transformer trained on source code as it is common practice nowadays in NLP tasks. The input structure is the same of the LSTM baseline. We used both the original pre-trained model and a variant available via the Transfomers library (Wolf et al., 2020) (alternative model link). Additionally, for this model we tested both fine-tuning and simple transfer learning.

PatchCodeBERT we used the pre-trained CodeBERT to build a twin version of it, replicating the initial model and feeding one with the added lines and one with the deleted lines.

Transformer we considered an uninitialised Transformer encoder with bi-directional attention (as BERT), thus re-proposing a smaller version of CodeBERT.

PatchTransformer similarly to what we did with the Transformer and CodeBERT, we used a smaller uninitialised version of PatchCodeBERT that we trained from scratch.

Since many of the models we considered use separate encoders for added and deleted lines in the patches, we developed a merging layer working on the intermediate hidden vectors. The proposed layer, similarly to the one employed by CC2Vec, concatenates the two vectors, their product, their difference, and their cosine and euclidean distances. The resulting vector is passed through a final classification layer. The remaining models directly apply the final projection on the hidden representation.

5 Experiments and results

Table 1: Results on the private data sets.

Method	$\mathbf{F_1}$	
	Validation	**Test**
XGBoost	0.692 ± 0.033	0.695
LSTM	0.823 ± 0.008	0.829
PatchRNN	0.696 ± 0.007	0.635
HAN	0.787 ± 0.007	0.777
CodeBERT	0.767 ± 0.019	0.764
PatchCodeBERT	0.731 ± 0.023	0.728
Transformer	$\mathbf{0.841 \pm 0.014}$	**0.870**
PatchTranformer	0.831 ± 0.014	0.827

Table 2: Results on the data set by Ponta et al. (2019).

Method	macro $\mathbf{F_1}$	
	Validation	**Test**
LSTM	0.661 ± 0.054	0.607
Transformer	$\mathbf{0.667 \pm 0.033}$	**0.635**
PatchTranformer	0.643 ± 0.020	0.601

We divided the experiments following the data sets division. First, we trained multiple models on the private data sets with the binary labelling system. We selected the best models from the first step for training on the third data set with the five macro-categories. To assess the goodness of the results we measured the F_1-*score* achieved by the classifiers on the test and validation sets. The F_1-*score* on the third data set is computed applying macro averaging among the macro-categories. Results on validation sets are reported as *avg.* $\pm$ *std* because we applied 3-fold cross validation.

We reported the results on the private data sets in Table 1. The transformer based solutions clearly outperformed the other models we considered. In this case we employed a 2 layers Transformer network with 32 hidden units, 4 attention heads, and a maximum of 768 tokens in the input sequence. Interestingly Transformer, LSTM, and PatchTransformer models, which achieved the best results, didn't undergo any pre-training, indicating that fine-tuning may be counterproductive in some cases.

We reported the results on the third data set in Table 2. Here we considered only the three best methods from the first experiment. The results confirmed those of the private data sets: the Transformer model performed better than all the other considered solutions. In this case we increased the hidden units size of the Transformer to 128. The drop in the F_1-*score* with respect to the previous experiment was expected since we moved to a multi-class problem where the issue of unbalance has most probably harmed the performances.

6 Conclusion

In this paper we evaluated different approaches for security patches detection in Java OSS using NLP technologies. Despite the general improvements in many NLP tasks due to the use of pre-trained models, in our experiments we found that uninitialised models yield better results than fine-tuned ones; this is most probably due to the insufficient presence of Java code in the pre-training of the considered models. Differently from previous works, we also noticed that using separate sub-models yields worse performances than using a single model.

In the future we are willing to work on two complementary directions. On one side we are interested in exploring other pre-trained models to refine, either sequential or graph ones. On the other side we are interested in working on larger data sets; thus exploiting C/C++ resources can be useful to produce improved models than can be then transferred to under-resourced languages like Java.

References

Miltiadis Allamanis, Earl T. Barr, Premkumar T. Devanbu, and Charles Sutton. 2018. A survey of machine learning for big code and naturalness. *ACM Comput. Surv.*, 51(4):81:1–81:37.

Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. 2019. code2vec: learning distributed representations of code. *Proc. ACM Program. Lang.*, 3(POPL):40:1–40:29.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. 2017. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146.

Leo Breiman. 2001. Random forests. *Mach. Learn.*, 45(1):5–32.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Minmin Chen. 2017. Efficient vector representation for documents through corruption. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794. ACM.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014,*

Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014, pages 103–111. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. Codebert: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1536–1547. Association for Computational Linguistics.

Abram Hindle, Earl T. Barr, Mark Gabel, Zhendong Su, and Premkumar T. Devanbu. 2016. On the naturalness of software. *Commun. ACM*, 59(5):122–131.

Thong Hoang, Hong Jin Kang, David Lo, and Julia Lawall. 2020. Cc2vec: distributed representations of code changes. In *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pages 518–529. ACM.

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

Marjan Hosseini, Alireza Javadian Sabet, Suining He, and Derek Aguiar. 2022. Interpretable fake news detection with topic and deep variational models. *CoRR*, abs/2209.01536.

Tae-Hwan Jung. 2021. Commitbert: Commit message generation using pre-trained programming language model. *CoRR*, abs/2105.14242.

Quoc V. Le and Tomás Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org.

Rocío Cabrera Lozoya, Arnaud Baumann, Antonino Sabetta, and Michele Bezzi. 2021. Commit2vec: Learning distributed representations of code changes. *SN Comput. Sci.*, 2(3):150.

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

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Matteo Muffo, Roberto Tedesco, Licia Sbattella, and Vincenzo Scotti. 2021. Static fuzzy bag-of-words: a lightweight and fast sentence embedding algorithm. In *4th International Conference on Natural Language and Speech Processing, Trento, Italy, November 12-13, 2021*, pages 176–185. Association for Computational Linguistics.

Matteo Muffo, Roberto Tedesco, Licia Sbattella, and Vincenzo Scotti. 2022. *Static Fuzzy Bag-of-Words: Exploring Static Universe Matrices for Sentence Embeddings*, pages 101–121. Springer International Publishing, Cham.

Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. 2018. A survey of the usages of deep learning in natural language processing. *CoRR*, abs/1807.10854.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. 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)*, pages 528–540. 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 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Serena Elisa Ponta, Henrik Plate, Antonino Sabetta, Michele Bezzi, and Cédric Dangremont. 2019. A manually-curated dataset of fixes to vulnerabilities of open-source software. In *Proceedings of the 16th International Conference on Mining Software Repositories, MSR 2019, 26-27 May 2019, Montreal, Canada*, pages 383–387. IEEE / ACM.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Abubakar Omari Abdallah Semasaba, Wei Zheng, Xiaoxue Wu, and Samuel Akwasi Agyemang. 2020. Literature survey of deep learning-based vulnerability analysis on source code. *IET Softw.*, 14(6):654–664.

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.

Steven Vaughan-Nichols. 2015. It's an open-source world: 78 percent of companies run open-source software.

Xinda Wang, Shu Wang, Pengbin Feng, Kun Sun, Sushil Jajodia, Sanae Benchaaboun, and Frank Geck. 2021. Patchrnn: A deep learning-based system for security patch identification. In *2021 IEEE Military Communications Conference, MILCOM 2021, San Diego, CA, USA, November 29 - Dec. 2, 2021*, pages 595–600. IEEE.

Xinda Wang, Shu Wang, Kun Sun, Archer L. Batcheller, and Sushil Jajodia. 2020. A machine learning approach to classify security patches into vulnerability types. In *8th IEEE Conference on Communications and Network Security, CNS 2020, Avignon, France, June 29 - July 1, 2020*, pages 1–9. IEEE.

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, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.

Jiajie Wu. 2021. Literature review on vulnerability detection using NLP technology. *CoRR*, abs/2104.11230.

Vitalii Zhelezniak, Aleksandar Savkov, April Shen, Francesco Moramarco, Jack Flann, and Nils Y. Hammerla. 2019. Don't settle for average, go for the max: Fuzzy sets and max-pooled word vectors. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yaqin Zhou, Jing Kai Siow, Chenyu Wang, Shangqing Liu, and Yang Liu. 2022. SPI: automated identification of security patches via commits. *ACM Trans. Softw. Eng. Methodol.*, 31(1):13:1–13:27.

Improving NL-to-Query Systems through Re-ranking of Semantic Hypothesis

Pius von Däniken[1] and **Jan Deriu**[1] and **Eneko Agirre**[2] and
Ursin Brunner[1] and **Mark Cieliebak**[1] and **Kurt Stockinger**[1]
[1] ZHAW Zurich University of Applied Sciences
{vode,deri,brrn,ciel,stog}@zhaw.ch
[2] HiTZ Center - Ixa, University of the Basque Country UPV/EHU
e.agirre@ehu.eus

Abstract

Natural Language-to-Query systems translate a natural language question into a formal query language such as SQL. Typically the translation results in a set of candidate query statements due to the ambiguity of natural language. Hence, an important aspect of NL-to-Query systems is to rank the query statements so that the most relevant query is ranked on top. We propose a novel approach to significantly *improve the query ranking and thus the accuracy of such systems*. First, we use existing methods to translate the natural language question (NL_{in}) into k query statements and rank them. Then we translate each of the k query statements back into a natural language question (NL_{gen}) and use the *semantic similarity* between the original question NL_{in} and each of the k generated questions NL_{gen} to *re-rank the output*. Our experiments on two standard datasets, OTTA and Spider, show that this technique improves even strong state-of-the-art NL-to-Query systems by up to 9 percentage points. A detailed error analysis shows that our method correctly down-ranks queries with missing relations and wrong query types. While this work is focused on NL-to-Query, our method could be applied to any other semantic parsing problems as long as a text generation method is available.

1 Introduction

NL-to-Query describes the task of translating natural language questions to meaningful representations, such as logical forms, executable code, or structured query languages like SQL. The application of neural networks and the introduction of larger datasets (Yin and Neubig, 2017; Yu et al., 2018; Brunner and Stockinger, 2021) has increased performance, but the task is far from solved.

Re-ranking of candidate query statements allows introducing additional information in the process (Yin and Neubig, 2019). For a given natural language question (NL_{in}), neural networks keep a beam search and produce k candidate query statements (QS). Our analysis shows that an oracle selecting the correct query among the top-scoring 15 candidates would improve the performance of publicly available systems by up to 10 accuracy points on the Spider benchmark (Yu et al., 2018).

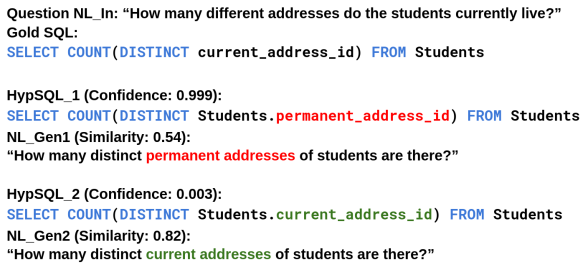


Figure 1: Example illustrates how semantic similarity is used to extract the correct hypothesis. `NL_In` is the input question, `Gold SQL` is the gold SQL query, `HypSQL_1` and `HypSQL_2` are generated by an NL-to-Query system (with confidence scores), and `NL_Gen1` and `NL_Gen2` are back-translated from the `HypSQL` statements, with scores by a similarity system. See text for further details.

Inspired by the success of back-translation in machine translation (Sennrich et al., 2016), we propose to *re-rank the candidate queries* according to the semantic similarity between the original question NL_{in} and the k synthetic questions NL_{gen} obtained via back-translating each of the k candidate queries into natural language. Figure 2 depicts the pipeline of our proposed system.

Figure 1 shows an example from the Spider dataset. For the question "How many different addresses do the students currently live?". The highest-ranked query according to the beam search ranking is HypSQL_1 with a confidence score of 0.999. However, this query returns the *permanent* addresses, which does not refer to the correct attribute, which would be the *current* addresses. In the example, the second hypothesis (i.e., HypSQL_2) has a much lower confidence of 0.003

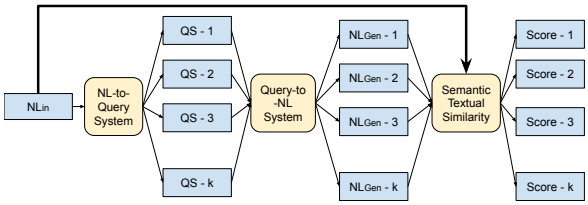


Figure 2: Pipeline of our system. NL_{in} = original natural language, QS = query statement, NL_{gen} = generated natural language.

although it fits the input question perfectly. On the other hand, the *semantic similarity* score between NL_{in} and the generated questions NL_{gen} shows a different picture: The back-translation of the correct hypothesis, i.e., NL_{gen2}, has a higher semantic similarity (0.82) than the back-translation of the incorrect hypothesis (0.54). Hence, semantic similarity would help to identify the correct query. This paper makes the following contributions:

- We present a novel method to improve NL-to-Query systems using re-ranking according to *Query-to-NL back-translation and semantic similarity*.
- We showcase improvements in two datasets using three systems, around $5-9$ points in OTTA (Deriu et al., 2020) and $2-3$ points in Spider (Yu et al., 2018).
- The error analysis shows that our method down-ranks hypotheses with missing relations or with incorrect query types.

2 Related Work

NL-to-Query (also referred to as Natural Language to Databases NLIDB) describes the task of translating natural language questions into structured queries (e.g., SQL). Most current approaches are based on sequence-to-sequence architectures (Yin and Neubig, 2017; Dong and Lapata, 2018; Suhr et al., 2018; Deriu et al., 2020), where the encoder is a recurrent neural network that generates a hidden representation of the natural language question, and the decoder is a recurrent neural network that generates the query. Alternatively, some approaches combine symbolic reasoning with information retrieval techniques (Sen et al., 2020). For a more in-depth treatment, we refer the reader to Affolter et al. (2019) and Odzcan et al. (2020).

In this work, we focus on the translation from natural language questions to database queries, where most recent approaches were proposed in the context of the text-to-SQL Spider dataset (Yu et al., 2018)[1]. Instead of working directly on SQL, some authors propose to use simpler and more general abstract syntax trees. For instance, Deriu et al. (2020) propose to use so-called Operation Trees, which we also used for this work.

Hypothesis Re-ranking is the task of creating an alternative ranking of k candidate solutions for a given task. The k candidates are usually the output of a beam search. In our case, the candidates are queries for the given natural language question. However, the problem of hypothesis re-ranking arises in many different generation tasks, not only NL-to-Query. For instance, Dušek and Jurcicek (2016) train a re-ranking network to score the generated hypotheses of their natural language generation model. Alternatively, (Deriu and Cieliebak, 2018; Agarwal et al., 2018) trained classifiers to predict the correctness of the hypotheses produced by their natural language generation system and select the hypothesis with the highest correctness score. Most of these approaches are developed in the field of natural language generation from structured data. For code generation, Yin and Neubig (2019) perform re-ranking by reconstructing the original utterance for the generated code. They use the reconstruction error as a measure for re-ranking. We are not aware of prior research on using textual semantic similarity to re-rank hypotheses in the field of NL-to-Query or Semantic Parsing in general.

Semantic Textual Similarity assesses to what degree two chunks of text are similar, usually on a 0-5 scale, which ranges from unrelated (0) to semantically equivalent (5) (Agirre et al., 2013). The advent of transformer-based models such as RoBERTa (Liu et al., 2019) has improved automatically assessing semantic textual similarity. Recently (Kane et al., 2020) introduced *NUBIA (NeUral Based Interchangeability Assessor for Text Generation)*. It extracts features from RoBERTa and GPT-2 (Radford et al., 2019) and fine-tunes a fully connected neural network to output a score between 0 and 1, indicating how interchangeable two input sentences are. Throughout this work, we will use *NUBIA* to automatically score the similarity between a natural question (NL_{in}) and a back-translated question (NL_{gen}).

Query-to-NL has the goal of translating a struc-

[1] https://yale-lily.github.io/spider

tured query into natural language and to provide a lay user with an explanation of the meaning of the query. A simple approach is to define production rules applied to the nodes of the abstract syntax tree (AST) of the query. Systems based on this idea have been developed for SQL (Koutrika et al., 2010), SPARQL (Ngonga Ngomo et al., 2013), Operation Trees (von Däniken, 2021), and queries expressed in lambda calculus (Wang et al., 2015). There are also systems based on neural networks such as (Xu et al., 2018). In this work, we leverage one of those systems to post-process the output of an NL-to-Query system. Others have also used query explanations to incorporate corrective feedback from the user in the NL-to-Query workflow (Elgohary et al., 2020; Labutov et al., 2018; Yao et al., 2019, 2020).

3 Method: Similarity for Re-ranking

The proposed method works in three steps (see also Figure 2): first, the NL-to-Query system translates the natural language input NL_{in} into a set of k candidate query statements QS - called our hypotheses. This is achieved by applying beam search during the decoding stage of a recurrent neural network. In the second step, each of the k hypotheses QS is translated back into natural language NL_{gen} using a Query-to-NL system. In the last step, each of the k back-translations NL_{gen} is compared to the original input using an off-the-shelf semantic textual similarity algorithm. We use the semantic similarity score to rank the hypotheses. For each NL_{in}, the top-scoring hypothesis is returned as the answer of the system.

3.1 Ranking Hypotheses based on Semantic Textual Similarity

Let NL_{in} be the user input (i.e., the natural language question) and $H = \{QS_1, ..., QS_k\}$ be the set of k hypotheses, i.e. candidate query statements QS_i, that are the output of the NL-to-Query system. In most cases, this set is the result of applying beam search for decoding. However, other approaches result in a set of hypotheses, for instance an ensemble of different NL-to-Query systems. In this work, we focus only on beam search-based hypothesis sets. Thus, each of the hypotheses has a confidence score c_i, which is used to rank the set of hypotheses, i.e., the candidate queries. We refer to this ranking as *Confidence*.

In a second step, each of the hypotheses QS_i is back-translated into a natural language question NL^i_{gen} using a Query-to-NL engine. Thus, we end up with a set of back-translated hypotheses $H_Q = \{NL^1_{gen}, ..., NL^k_{gen}\}$.

In a third step, we compute for each back-translated hypothesis the semantic textual similarity score with the user input NL_{in}, i.e., $s_i = \mathit{SemSim}(NL_{in}, NL^i_{gen})$. The set of hypotheses can be ranked according to the semantic similarity scores. We refer to this ranking as *Semantic*.

3.2 Weighting Strategies

Since the two rankings, *Confidence* and *Semantic* may disagree on the top hypothesis in some cases (as we have shown in the example in Figure 1), we combine the two scores c_i and s_i into a new ranking. For this, we propose the following weighting strategies:

Equal Weighting. The naive strategy is based on simply multiplying the two scores, that is $m_i^{equal} = c_i * s_i$, and we rank the set of hypotheses according to m_i^{equal}. We refer to this ranking as *Equal Weighting*.

Calibrated Weighting. Since the confidence scores and the semantic similarity scores have different distributions, the influence of each score in the *Equal Weighting* is not equal. For instance, in some cases, the influence of c_i is stronger than s_i and vice-versa. To counteract this effect, we decided to *calibrate both scores before multiplying*. A calibrated score should reflect the proportion of correct hypotheses selected, e.g., when a calibrated system assigns a score of 0.8 to a hypothesis, this hypothesis will be correct in 80% of the cases.

We use *Platt Scaling* (Platt, 2000) to calibrate both scores. This works by training a logistic regression model on the outputs of a model to transform these outputs into probability distributions. More precisely, for the confidence scores and the semantic scores respectively, a logistic regression model is trained. For this, we have to set aside a few hypotheses (more details later on). For the confidence calibration, a logistic regression model is trained on a set of pairs of confidence score and a label that indicates if the query is correct, i.e., $\mathbf{D} = \{(c_i, I^i_{\text{corr}}\}$. Analogously, we train a logistic regression model for the semantic similarity score s_i. Thus, the calibrated scores can be interpreted as the probability of the query being correct, i.e., $c_i^{calib} = Pr(I^i_{\text{corr}} = 1|c_i)$ and $s_i^{calib} = Pr(I^i_{\text{corr}} = 1|s_i)$. We call the score after

calibration c_i^{calib} and s_i^{calib} and the resulting mixed score $m_i^{calib} = c_i^{calib} * s_i^{calib}$. The resulting ranking is called *Calibrated Weighting*.

Learned Weighting. A natural extension of the calibration idea is to *train a logistic regression on both scores* at the same time, instead of independently. That is, we train a logistic regression model on pairs of confidence and semantic-scores[2], i.e., $\mathbf{D} = \{((c_i, s_i), I^i_{\text{corr}}\}$. This way, the model can learn the mixed proportions directly. Thus, $m_i^{learned} = Pr(I^i_{\text{corr}} = 1|c_i, s_i)$. For this, we again have to set aside a few hypotheses. We use the predicted probabilities from the logistic regression model to rank hypotheses and call the resulting ranking *Learned Weighting*.

Threshold Weighting. We observed that the confidence scores c_i are high in most cases in which the *Confidence* ranking yields a correct query. In many cases where the *Confidence* ranking yields wrong queries, the confidence scores are low. However, the *Semantic* scores tend to be higher. Thus, we propose the following strategy: If the maximum confidence score of the hypotheses set is above a threshold, we use the *Confidence* ranking, otherwise, we use the *Semantic* ranking. We refer to this ranking as *Threshold Weighting*. The threshold is calculated by first determining the 90th percentile over the confidence scores of all training hypotheses and then finding the lowest confidence of a correct hypothesis that lies above that.

Upper Bounds. To determine the theoretical upper bounds of our approach, we introduce two oracles. The first oracle selects the correct hypothesis from the candidates if there is one. The second oracle selects the correct hypothesis between the two top-ranked hypotheses by *Confidence* or *Semantic* if there is one. The first oracle determines the potential of re-ranking in general (we refer to it as *Oracle*). The second oracle determines the maximum contribution that the semantic similarity could do to Confidence (we refer to it as *Oracle-Sem*).

4 Experimental Setup

In this section, we describe the experimental setup, the datasets, the NL-to-Query models, the Query-to-NL model, and the semantic textual similarity model.

[2]Using more features, e.g., the length of the generated query or m_i^{equal} did not yield any improvements.

4.1 Datasets

We analyzed our approach on two different datasets used as benchmarks for evaluating NL-to-Query systems: Spider (Yu et al., 2018) and OTTA (Deriu et al., 2020). Both datasets contain complex queries and cover large amounts of attributes of the databases. Spider contains around 10K queries against 200 different databases. The dataset is used to study NL-to-SQL translations. OTTA contains around 3.8K queries over 5 databases. OTTA is used to study translations from NL-to-OT (Operation Trees) which are similar to abstract syntax trees (AST), i.e., an intermediate query language can be translated to other query languages such as SQL or SPARQL. OTTA contains more complex queries with longer join paths than Spider. From the OTTA corpus, we used only queries against the databases *Moviedata* and *Chinook* since they contain the largest amounts of queries. Details about the queries used for each dataset are given below.

4.2 NL-to-Query Models

We applied publicly available machine learning models trained for the datasets, which produce queries with filter values in the WHERE-clauses as otherwise there would be placeholder tokens in the back-translations. For all models, we use a beam size[3] of $k = 15$. For the OTTA corpus, we used the pre-trained *GrammarNet* by (Deriu et al., 2020). The output of *GrammarNet* is a set of Operation Tree (OT) hypotheses, which represent the query. OTs can be translated to SQL and executed on an SQL database. For each of the two domains in OTTA (i.e., *Moviedata* and *Chinook*), we use a specifically trained *GrammarNet*. We refer to these models as *GrammarNet-Moviedata* and *GrammarNet-Chinook*. For the Spider dataset, we apply two strong NL-to-SQL systems that are publicly available. The first system is *BridgeV2* (Lin et al., 2020), which returns a set of hypothesis SQL queries from a beam search decoder. We refer to this model as *Spider-BridgeV2* [4]. The second system is *ValueNet* by Brunner and Stockinger (2021), which also returns a set of SQL hypotheses from a beam search decoder[5]. We refer to this model as

[3]In preliminary experiments, we noted that using a larger beam size does not impact the scores significantly.

[4]We chose these systems for their strong performance, code availability and quality of code.

[5]The API provided by the authors included confidence scores based on the sum per-token-confidence instead of average. We approximated the average by dividing the provided score by the number of characters in the SQL hypothesis.

Spider-ValueNet.

4.3 Query-to-NL Model

For back-translating queries to natural language, we use the *Operation Tree-to-Text (OT3)* system kindly made available by von Däniken (2021). It translates OTs into natural language questions in a rule-based manner, which ensures that most OTs are translated correctly, i.e., no nodes are left out or added during translation. OT3 is domain-agnostic, which allows it to be easily adapted to a new domain by just defining domain-specific metadata, i.e., the canonical names of the tables, attributes and types. The main advantage of OT3 is the ability to express relationships naturally, which results in more fluent back-translations. There are currently some limitations with the state-of-the-art Query-to-NL models, which do not handle more complex constructs [6] well. Thus, we perform the evaluation only on the queries that are handled by OT3. More details can be found in Appendix A.

4.4 Semantic Textual Similarity Model

In order to compute textual semantic similarity between two questions, i.e., between NL_{in} and NL_{gen}, we apply NUBIA (Kane et al., 2020), a pre-trained model that scores a pair of sentences based on their interchangeability. We use NUBIA[7] out-of-the-box without any fine-tuning.

4.5 Mixed Strategies

For the *Calibrated Weighting*, the *Learned Weighting*, and the *Threshold Weighting* rankings, labeled data points are needed for setting up the mixed strategy. The samples are used to train the logistic regression models for the *Calibrated Weighting* and the *Learned Weighting*. We use the implementation provided by *scikit-learn* (Pedregosa et al., 2011) with balanced class weights and all other parameters as default. For the *Threshold Weighting*, these samples are used to determine the threshold for when to select the *Confidence* ranking or the *Semantic* ranking. We use k-fold cross-validation with k[8] chosen such that there are 20 samples in each fold[9] for training the strategies for each split. We report accuracies averaged over the k test splits for all strategies.

[6] E.g., GroupBy, SetOperations, or Nested Quieries

[7] https://github.com/wl-research/nubia

[8] Concretely, $k = 22$ for Spider, $k = 11$ for Moviedata, and $k = 12$ for Chinook.

[9] This results in $20 * 15 = 300$ data points for training the logistic regression models.

5 Results

As explained in the previous section, we evaluate the effectiveness of our approach over two different datasets consisting of 22 databases using three different systems, as shown in Table 1. We evaluate the systems using the *component equality* proposed by Yu et al. (2018). We can see that for all datasets one of our *re-ranking approaches outperforms the baseline* without re-ranking up to 9%. We will now analyze our re-ranking approaches in more detail.

Semantic Re-ranking. In all cases, except for *Chinook*, the *Semantic-based* re-ranking performs worse than the baseline system ranking (Confidence), showing that our method alone has not enough information to select the correct hypothesis.

Mixed Re-ranking (i.e. Equal, Calibrated, Learned, Threshold). On the contrary, the combination of the Confidence and Similarity scores improves over *Confidence* alone in all mixed strategies (with a minor exception for Threshold for ValueNet in Spider). The improvement ranges from $2-3\%$ on *Spider* to $5-9\%$ on *OTTA*. These results show that our method injects new information and improves over the base systems. In all cases, the simple *Equal Weighting* performs well, making it a great default mixed strategy. The results or other mixed strategies are better in some cases, although the best mixed strategy varies in each column. For instance, for *Spider-Bridge* the *Threshold Weighting* strategy works best, yielding an improvement of 2.56 points in accuracy.

Oracle. The difference between *Confidence* and *Oracle*, i.e. the optimal re-ranking, lies at around 18% for both *OTTA* subcorpus and $8-10\%$ for Spider, depending on the system. The differences in margins between *Spider* and *OTTA* can be explained by the fact that the *Spider*-based models achieve higher *Confidence* accuracies, which decreases the margin for improvement.

Oracle-Sem. The difference between the best mixed strategy and *Oracle-Sem* is around 3 points. Thus, there is a potential improvement of around 3 points left for all systems using semantic similarity. However, the difference between the *Oracle-Sem* score and the *Oracle* score differs between the *Spider*-based systems and the *OTTA*-based systems. While the difference in the *Spider*-based systems is between 3 to 4 points, the difference for the *OTTA*-based systems is between 6 to 7 points.

Dataset System	OTTA-Chin. GrammarNet	OTTA-Movie GrammarNet	Spider Bridge	Spider ValueNet
Confidence	42.89	52.24	71.46	74.31
Semantic	48.16 (+5.27)	45.25 (−6.99)	62.70 (−8.76)	68.01 (−6.30)
Equal	**51.84** (+8.95)	59.23 (+6.99)	73.03 (+1.57)	76.83 (+2.52)
Calibrated	51.44 (+8.55)	**59.60** (+7.36)	73.78 (+2.32)	**77.22** (+2.91)
Learned	51.48 (+8.59)	59.44 (+7.20)	73.93 (+2.47)	77.09 (+2.78)
Threshold	46.90 (+4.01)	54.71 (+2.47)	**74.05** (+2.59)	71.30 (−3.01)
Oracle-Sem	54.94 (+12.05)	62.38 (+10.14)	77.30 (+5.84)	80.35 (+6.04)
Oracle	61.32 (+18.43)	69.98 (+17.74)	81.12 (+9.66)	83.12 (+8.81)

Table 1: Accuracy of our approach for translating NL questions to OTs and SQL, respectively, using three different systems and two different datasets. The deltas with respect to the *Confidence* ranking (baseline) are shown in parentheses. *Oracle-Sem* and *Oracle* are theoretical upper bounds.

6 Discussion

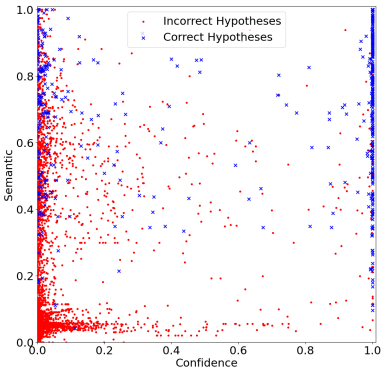
Based on the results, we see that including semantic similarity for re-ranking works better than using the *Confidence* scoring only. In this section, we explore the potential and limitations of this approach in more detail.

6.1 Confidence Score vs. Semantic Similarity Score

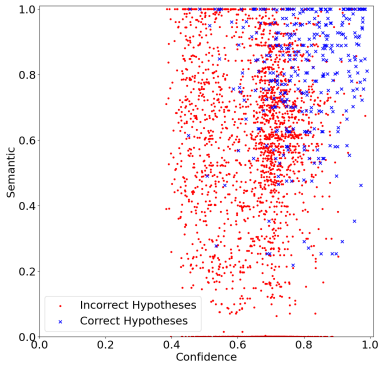
To better understand the results, we analyze the relationship between the confidence scores and the semantic scores. In Figure 3, the confidence scores are plotted against the semantic similarity scores, where blue dots denote correct hypotheses, and red dots denote incorrect ones. We perform the analysis on the *Bridge* system over Spider and the GrammarNet system over *Moviedata*, as they show the clearest difference in score distributions.

First, we note that the distributions for the two systems look different. For *Bridge* the confidence scores mostly lie at the edges, either at 0.0 or 1.0. The *Moviedata* confidence scores are more evenly distributed between 0.4 and 1.0. On the other hand, the semantic similarity scores are evenly distributed in both cases.

Second, we note that for the *Bridge* system, confidence scores close to 1.0 are reliable, i.e., a hypothesis with confidence close to 1.0 tends to be correct. On the other hand, correct hypotheses with low confidence tend to have higher semantic scores (see upper left corner). This explains the strong performance of *Threshold Weighting* for *Bridge*. For *Moviedata*, the picture is different. The correct samples tend to have both high confidence and high semantic scores (upper right corner). Thus, the other weighing strategies tend to perform well, while *Threshold Weighting* under-performs.



(a) Spider-BridgeV2



(b) Moviedata-GrammarNet

Figure 3: Confidence scores and semantic similarity scores for hypotheses produced by *Spider-BridgeV2* and *Moviedata-GrammarNet*. Every cross corresponds to a hypothesis. Blue indicates correct hypotheses and red incorrect ones.

Third, we note that semantic scoring alone is not sufficient. For *Bridge*, the semantic score tends to score correct hypotheses as low as the incorrect ones (see lower part). However, it works well for finding incorrect hypotheses. Although the distributions for *Bridge* and *Moviedata* have great differences, our approach works in both cases.

Error Type	Missing Join			
Original Question	List all singer names in concerts in year 2014.			
Ranking	**SQL**	**Back-translated Question**	c_i	s_i
Gold	SELECT T2.name FROM singer_in_concert AS T1 JOIN singer AS T2 ON T1.singer_id = T2.singer_id JOIN concert AS T3 ON T1.concert_id = T3.concert_id WHERE T3.year = 2014	What are the names of singers who performed in concerts whose year is 2014?	-	-
Baseline	SELECT singer.Name FROM singer_in_concert JOIN singer ON singer_in_concert.Singer_ID = singer.Singer_ID WHERE singer.Song_release_year = 2014 (**missing table "concert"**)	What are the names of singers who were released in 2014 who performed in concerts?	0.020	0.792
Semantic	SELECT singer.Name FROM singer_in_concert JOIN singer ON singer_in_concert.Singer_ID = singer.Singer_ID JOIN concert ON singer_in_concert.concert_ID = concert.concert_ID WHERE concert.Year = 2014	What are the names of singers who performed in concerts whose year is 2014?	0.015	0.823

Error Type	Wrongly added Filter			
Original Question	Find the pixel aspect ratio and nation of the tv channels that do not use English.			
Ranking	**SQL**	**Back-translated Question**	c_i	s_i
Gold	SELECT Pixel_aspect_ratio_PAR , country FROM tv_channel WHERE LANGUAGE $\neq$ 'English'	What are the aspect ratios and countries of tv channels whose language is not English?	-	-
Baseline	SELECT TV_Channel.Pixel_aspect_ratio_PAR, TV_Channel.Country FROM TV_Channel WHERE TV_Channel.Language $\neq$ "English"	What are the aspect ratios and countries of tv channels whose language is not english?	1.000	0.654
Semantic	SELECT TV_Channel.Pixel_aspect_ratio_PAR, TV_Channel.Country FROM TV_Channel WHERE TV_Channel.Language $\neq$ "English" AND TV_Channel.Country $\neq$ "English" (**wrong additional filter**)	What are the aspect ratios and countries of tv channels whose country is not english and whose language is not english?	0.008	0.673

Table 2: Examples of types of errors due to re-ranking. For each error type, we show the natural language question and the corresponding SQL gold standard. Next we show the top candidates according to the *Confidence* ranking and the *Semantic* ranking. c_i and s_i refer to confidence score of the NL-to-query translation and the similarity score between the natural language questions, respectively.

NL_{in}:	Whats the average track size of tracks purchased from 120 S Orange Ave?				
i	NL_{gen}	c_i	s_i	m_i^{equal}	OK
1	What is the average size of all tracks on invoice lines which are part of invoices?	0.669	0.49	0.327	F
2	What is the average size of all tracks on invoice lines which are part of invoices whose billing street is 120 S Orange Ave?	0.668	0.61	0.407	T
9	What is the average size of all tracks on Albums on invoice lines which are part of invoices whose billing street is 120 S Orange Ave?	0.632	0.3	0.1896	F
NL_{in}:	Which companies from Mexico produced their films in Mexico ?				
i	NL_{gen}	c_i	s_i	m_i^{equal}	OK
1	What are the names of companies which produced movies whose status is Mexico?	0.729	0.676	0.492	F
3	What are the names of companies which produced movies which were produced in countries whose name is Mexico?	0.712	0.751	0.534	T
5	What are the names of companies which produced movies whose name is Mexico?	0.664	0.741	0.492	F
NL_{in}:	What are the distinct template type descriptions for the templates ever used by any document?				
i	NL_{gen}	c_i	s_i	m_i^{equal}	OK
1	What are the distinct descriptions of template types for templates?	0.494	0.686	0.338	F
2	What are the distinct descriptions of template types for templates used for documents?	0.091	0.973	0.166	T
3	Show me everything about template types.	0.031	0.133	0.050	F

Table 3: Illustrative examples of the impact of re-ranking. We show three original questions (NL_{in}) and the corresponding back-translated examples (NL_{gen}). Value i denotes the rank in the *Confidence* ranking, c_i is the confidence score of the decoder, s_i is the similarity score, m_i^{equal} is the combination of c_i and s_i, OK indicates whether the generated query is correct (T = true, F = false).

6.2 Error Analysis: Confidence vs. Semantic Ranking

To better understand the differences between the *Semantic* and *Confidence* rankings, we analyze the cases in which one of the two ranking schemes returns a correct query, and the other one does not. This analysis is performed on the *Bridge* output where in 19.2% of the cases, only one of the two ranking schemes returns the correct hypothesis. In 25% of the cases in which only the *Confidence*

ranking returns a correct query, the *Semantic* ranking returned a query with a redundant *WHERE*-clause, and in 20% of cases, the *Semantic* ranking returned a wrong attribute in the projection. This suggests that the *Semantic* ranking is not stable against redundant information in the query and slight variations in the return attributes.

In the cases where only the *Semantic* ranking returns a correct query, the query returned by the *Confidence* ranking contains missing or redundant *Join*-clauses in 47% of cases and wrong query types in 21% of cases. This suggests that the *Semantic* ranking's strength lies in detecting missing relations and detecting wrong query types (i.e., SUM instead of COUNT).

In Table 2 two examples of errors are shown. The first example shows a missing join operation of *Confidence*. In particular, the table "concert" is missing in the SQL statement. In this case the confidence score of the wrong *Confidence* query, i.e. $c_i = 0.02$, is higher than the confidence of the correct *Semantic* query, i.e. 0.015. On the other hand, the semantic textual similarity score s_i of the correct *Semantic* query, i.e., 0.823, is higher than the score of the incorrect *Confidence* query, i.e., 0.792. We note that although the confidence score of the incorrect query is the highest of all hypotheses, it is a low score. Usually, the confidence scores are around 1.0.

The second example shows the problem of an additional filter (TV_Channel.Country $\neq$ "English"), which confuses the semantic similarity score. The *Confidence* ranking selects the correct query with high confidence, i.e. 1.0. However, the semantic score of the incorrect *Semantic* query, i.e., 0.673, is higher than the semantic score of the correct query, i.e., 0.654.

This phenomenon motivates the *Threshold Weighting*. The reason is that high confidence scores from the NL-to-Query system are more trustworthy than the semantic scores. However, in cases where the NL-to-Query system is not confident, the semantic score performs well. The automatically determined threshold in our experiments lies at around 0.9.

6.3 Qualitative Analysis

In Table 3, we show examples of the different rankings. We show three representative examples of a 15-best list. In the first example, we note that the hypothesis with the best confidence score, i.e., c_1 = 0.669, is incorrect. The second best hypothesis, according to the confidence score, is correct and has a very similar score to the hypothesis placed first (0.669 *vs.* 0.668). The hypothesis that is placed 9^{th} adds an unnecessary relation. However, the confidence score is still close to the hypothesis placed first. The semantic score, on the other hand, is more accurate. The correct hypothesis is placed 1^{st} with a large margin (0.61 *vs.* 0.49) and an even larger difference with the score of the 9^{th} place. Finally, the combined score m_2^{equal} of 0.407 clearly identifies result 2 as the correct one.

The second example shows a similar pattern: the first hypothesis with a confidence score c_1 of 0.729 is obviously wrong. The second hypothesis, which is correct, has a slightly lower confidence score c_2 of 0.712. The *Semantic* score s_3 of 0.751 ranks the set of hypotheses correctly. However, *Semantic* re-ranking alone is not enough since the 5^{th} ranked example has a very high semantic similarity score while being incorrect. In this case the *Equal Weighing* approach m_i^{equal} helps differentiating: While s_3 and s_5 are very close, m_3^{equal} and m_5^{equal} have a bigger margin.

The last example shows a case where the *Equal Weighting* does not work. Although the semantic score s_2 of 0.933 works to find the correct answer, the confidence score c_2 of 0.091 of the correct hypothesis is much lower than the confidence of the incorrect hypothesis, $c1$ of 0.494. In this case, the *Threshold Weighting* would work well as it relies on s_i for the cases where the maximum confidence score is too low.

7 Conclusion

We proposed a novel approach to improve semantic NL-to-Query systems based on back-translating the generated query into a natural language question, and re-ranking the top hypothesis of the NL-to-Query system according to the semantic similarity of the generated questions with regard to the original question. Our approach improves over strong, publicly available systems by up to 3 percentage points on the Spider dataset and up to 9 points on the OTTA dataset.

Our results clearly show the potential of back-translation for improving NL-to-Query systems, and it could be applied to more general semantic parsing problems as long as a generation method is available.

Acknowledgements

This work was supported by the European Union's Horizon 2020 research and innovation program under grant agreement No. 863410.

References

Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*, 28(5):793–819.

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.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Angela Bonifati, Wim Martens, and Thomas Timm. 2017. An analytical study of large SPARQL query logs. *Proc. VLDB Endow.*, 11(2):149–161.

Ursin Brunner and Kurt Stockinger. 2021. Valuenet: A natural language-to-sql system that learns from database information. *International Conference on Data Engineering (ICDE)*.

Jan Deriu, Katsiaryna Mlynchyk, Philippe Schläpfer, Alvaro Rodrigo, Dirk von Grünigen, Nicolas Kaiser, Kurt Stockinger, Eneko Agirre, and Mark Cieliebak. 2020. A methodology for creating question answering corpora using inverse data annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 897–911, Online. Association for Computational Linguistics.

Jan Milan Deriu and Mark Cieliebak. 2018. Syntactic manipulation for generating more diverse and interesting texts. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 22–34.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. 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.

Ahmed Elgohary, Saghar Hosseini, and Ahmed Hassan Awadallah. 2020. Speak to your parser: Interactive text-to-SQL with natural language feedback. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2065–2077, Online. Association for Computational Linguistics.

Hassan Kane, Muhammed Yusuf Kocyigit, Ali Abdalla, Pelkins Ajanoh, and Mohamed Coulibali. 2020. Nubia: Neural based interchangeability assessor for text generation.

Andreas Kokkalis, Panagiotis Vagenas, Alexandros Zervakis, Alkis Simitsis, Georgia Koutrika, and Yannis Ioannidis. 2012. Logos: A system for translating queries into narratives. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, page 673–676, New York, NY, USA. Association for Computing Machinery.

G. Koutrika, A. Simitsis, and Y. E. Ioannidis. 2010. Explaining structured queries in natural language. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 333–344.

Igor Labutov, Bishan Yang, and Tom Mitchell. 2018. Learning to learn semantic parsers from natural language supervision. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1676–1690, Brussels, Belgium. Association for Computational Linguistics.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, November 16-20, 2020*.

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. *CoRR*, abs/1907.11692.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak sparql: Translating sparql queries into natural language. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 977–988, New York, NY, USA. Association for Computing Machinery.

Fatma Odzcan, Abdul Quamar, Jaydeep Sen, Chuan Lei, and Vasilis Efthymiou. 2020. State of the art and open challenges in natural language interfaces to data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 2629–2636, New York, NY, USA. Association for Computing Machinery.

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.

John Platt. 2000. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv. Large Margin Classif.*, 10.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jaydeep Sen, Chuan Lei, Abdul Quamar, Fatma Özcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. 2020. Athena++: Natural language querying for complex nested sql queries. *Proc. VLDB Endow.*, 13(12):2747–2759.

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.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. 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 2238–2249, New Orleans, Louisiana. Association for Computational Linguistics.

Pius von Däniken. 2021. Improving a semantic parser through user interaction. *Publikationen School of Engineering*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China. Association for Computational Linguistics.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. SQL-to-text generation with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 931–936, Brussels, Belgium. Association for Computational Linguistics.

Ziyu Yao, Yu Su, Huan Sun, and Wen-tau Yih. 2019. Model-based interactive semantic parsing: A unified framework and a text-to-SQL case study. 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 5447–5458, Hong Kong, China. Association for Computational Linguistics.

Ziyu Yao, Yiqi Tang, Wen-tau Yih, Huan Sun, and Yu Su. 2020. An imitation game for learning semantic parsers from user interaction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6883–6902, Online. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2019. Reranking for neural semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4553–4559, Florence, Italy. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

A On Query-to-NL

While Query-to-Text is not a contribution of our work, we discuss and motivate our choice of OT3 as our Query-to-Text engine. We adapted OT3 to handle all the domains in the Spider development set, which comprises 20 databases. In order to handle SQL queries, we translate SQL queries into OTs using a rule-based approach. The main advantage over statistical methods is that we can be sure that the queries are correctly back-translated to text. This is due to the rule-based nature of OT3.

Sanity Check. In order to show that OT3 correctly renders the semantics of a query, we first perform a sanity check, where we backtranslated the gold-standard tree for a given question. Thus, we need to show that the original question and the back-translation are semantically equivalent. As negative examples, we also mix in randomly sampled human questions, thus the original question and the negative back-translation should never be semantically equivalent. We let humans annotate this data, that is, we showed humans pairs of original questions and either a positive or negative back-translation. In this setting, humans agree in 94% of cases with the parsing ground-truth. This shows that the synthetic questions are understandable and

generally maintain the semantics of the underlying OT. The experiments show that the synthetic questions are of high quality and can be used as basis for re-ranking.

Limitations. OT3 does not handle *GROUP BY*, *sub-queries* and *set operations*, thus, we discard these samples from the Spider and OTTA development sets, keeping 82% of *OTTA-Moviedata*, 76% of *OTTA-Chinook* and 43% of *Spider*. The reported results are on these subsets of the datasets. Note that several studies on natural language query logs (Bonifati et al., 2017; Affolter et al., 2019) show that typical queries in real-world applications are far less complex than the ones contained in the Spider dataset. Hence, not supporting GROUP BYs, sub queries or set queries is not a significant issue in a real-world scenario. Note that our method can still be applied to the full datasets, defaulting to the Confidence ranking when none of the hypotheses could be back-translated. The positive results are consistent, but the improvement is lower, correlated with the coverage. E.g. an overall improvement of 0.67% for the whole Spider (with the Bridge system using equal mixed re-ranking), which roughly corresponds to the 1.57% improvement obtained on the 43% subset of Spider which does not contain complex SQL operations.

Selection. The choice of OT3 is motivated by the fact that it renders relationships between entities naturally. For instance, the relationship between persons and movies, which is modelled via the cast table, is expressed as "Persons that play in movies". For instance, Logos (Kokkalis et al., 2012) expresses the same relationship as "Persons associated with movies", which is not natural and cannot be handled by our semantic textual similarity tool. We also evaluated statistical models, which suffer from hallucinations (i.e., adding text that is not semantically related to the query) and are generally unreliable. Thus, we are not aware of any Query-to-Text solution, that handles all types of queries (Group By, Set Operations, Nested Queries) such that the generated texts read naturally. Thus, OT3 has proved to be best suited for our task.

B On Evaluation

We adapted the *Component Equality* measure for operation trees (OTs) since we translate the SQL queries of the *Spider*-based systems to OTs. For OTs, this measure checks if the nodes of the predicted tree correspond to the nodes of the gold standard tree. This allows measuring query equality independently of the order of the nodes. Furthermore, we adapted this analysis also to measure if the *Join* attributes are rendered correctly. We decided against a result-based evaluation since it is impossible to reasonably evaluate queries that return an empty result set, often leading to over-estimating the quality of NL-to-Query systems. This happens often in cases where the result set is empty or for count questions. For Spider the databases are very small and do not contain much data, thus, queries tend to return empty results. For OTTA, which uses Yes/No questions, this problem is even more pronounced. Thus, the result-based evaluation is not reliable, and we opted for the component-based evaluation, which is now the standard evaluation for the Spider dataset.

Experimenting with ensembles of pre-trained language models for classification of custom legal datasets

Tamara Matthews and **David Lillis**
School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland
tamara.matthews@ucd.ie; david.lillis@ucd.ie

Abstract

Document corpora owned by law and regulatory firms pose significant challenges for text classification; being multi-labelled, highly imbalanced, often having a relatively small number of instances and a large word count per instance. Deep learning ensemble methods can improve generalization and performance for multi-label text classification but using pre-trained language models as base learners leads to high computational costs.

To tackle the imbalance problem and improve generalization we present a fast, pseudo-stratified sub-sampling method that we use to extract diverse data subsets to create base models for deep ensembles based on fine-tuned models from pre-trained transformers with moderate computational cost such as BERT, RoBERTa, XLNet and Albert. A key feature of the sub-sampling method is that it preserves the characteristics of the entire dataset (particularly the labels' frequency distribution) while extracting subsets. This sub-sampling method is also used to extract smaller size custom datasets from the freely available LexGLUE legal text corpora. We discuss approaches used and classification performance results with deep learning ensembles, illustrating the effectiveness of our approach on the above custom datasets.

1 Introduction

The increasing volume of regulations relating to activities in the law and regulation domain require efficient methods for automated multi-label classification (MLC), which can replace expensive and time-consuming data labelling by domain experts. Examples include legal professionals who may need to categorize the types of business activities a contract relates to, or label types of individual clauses (e.g. restrictive covenants, penalty clauses). Similarly, companies involved in financial regulation may need to label regulatory texts with the focus of the regulation (e.g. fraud, accounting obligations or mis-selling).

Since the creation of BERT (Devlin et al., 2018) the "foundation model" for pre-trained transformer-based language models, various new types of transformer models for NLP have been developed: some improve the pre-training method such as RoBERTa (Liu et al., 2019) and DeBERTa (He et al., 2020), the vocabulary size and specificity such as LEGAL-BERT (Chalkidis et al., 2020), or enable input sequence lengths larger than 512 tokens (Tay et al., 2020a) such as Reformer (Kitaev et al., 2020), CogLTX (Ding et al., 2020), Longformer (Beltagy et al., 2020), Big bird (Zaheer et al., 2020).

The major advantage of using pre-trained transformers for MLC is their large, context and semantic aware language models (LM), which can significantly improve classification results even on small datasets (Sun et al., 2019). Fine-tuning pre-trained language models on legal texts can be challenging due to their uncommon vocabulary (containing domain specific, rare, and conceptually complex words) and the large text length in each document, exceeding the maximum sequence input length of pre-trained transformer-based models.

There is a trade-off between the pre-trained model size and the available computational power. The pre-trained model size increases with the vocabulary size and with the number of parameters (the number of bi-LSTM layers). The larger models enable improved performance at the cost of increased computational effort (memory size and processing units) (Tay et al., 2020b).

While pre-trained transformer models can provide efficient solutions for text classification, fine-tuned models often lack flexibility as the fine-tuning restricts generalization to the new (narrower) domain. MLC based on deep ensemble models can be improve generalization but these approaches are constrained by high computational costs.

Here we present MLC results using deep-learning ensembles of fine-tuned models discussing

their suitability for limited resources environments and small size datasets (6000 to 7000 instances). Our approach is based on ensembles of fine-tuned models obtained by transfer learning from pre-trained transformers. We use transformers that require moderate fine-tuning costs as BERT, RoBERTa, XLNet (Yang et al., 2019) and ALBERT (Lan et al., 2020). This study includes homogeneous and heterogeneous ensembles of deep base-learners each generated from data re-sampling with replacement, using our custom pseudo-stratified random sampling method.

Our contribution includes: a custom pseudo-stratified sampling method for sub-sampling and train/test splitting of imbalanced multi-label datasets, used for generating diverse datasets; aggregating ensemble models using fine-tuned base-models (transfer learning from pre-trained NLP transformers), and discussing results. The datasets used in our work, sub-sampled from benchmark legal text datasets are available.

2 Multi-label classification

Real-life datasets as multi-domain business documentation or collections of legal and regulatory documents are multi-labelled and highly imbalanced, presenting a complex MLC problem, relating one example to multiple categories.

The imbalanced label distribution increases the complexity of the learning problem as stratified sampling cannot be applied for creating a balanced test/train split that includes all labels. Iterative methods for test-train split have been proposed to ensure a similar label distribution in both test and train sets (Sechidis et al., 2011).

Several approaches are designed to enable classic algorithms (Naive Bayes, SVM, Random Forests, k-means) to perform on multi-label datasets: (a) Algorithm Adaptation (Szyma, 2019) (b) Problem Transformation (Binary Relevance, Label Powerset and Classifier Chains methods) and (c) Ensemble learning, described in recent reviews dedicated to MLC (Bogatinovski et al., 2022; Kowsari et al., 2019). Since neural networks and deep learning methods can generate a multiple prediction output by design, such methods support multi-label classification and can be applied with or without approaches (a)-(c), although these can significantly improve the performance of deep learning methods also.

The multi-label data can produce ensembles of models, using one-vs-all and one-vs-one techniques. Other methods are based on re-sampling and sub-sampling (Bagging and Boosting) or re-arranging the data into convenient domains matching labels' distribution: random k-labelsets (Tsoumakas and Vlahavas, 2007), hierarchical arrangements, pruned sets (Read et al., 2008).

2.1 Performance measures and thresholding

A variety of performance measures have been designed to assess the various goals in multi-label classification: Hamming loss, ranking loss, one-error, average precision, coverage, micro-F1 and macro-F1. Used for imbalanced datasets, F-scores (measures) can be optimized either as an empirical utility maximization EUM (optimal classifier) or as a decision-theoretic approach DTA (predictions by optimal classifiers) (Lewis, 1995), DTA being better for handling rare classes and for domain adaptation tasks.

The F-measure optimization is usually performed in two steps: learn a score function from a ML algorithm (optimized for DTA) and then select a threshold to maximize the empirical F-measure (Ye et al., 2012).

3 Ensemble deep learning

Ensemble learning methods (Madjarov et al., 2012; Read et al., 2008; Tsoumakas and Vlahavas, 2007; Dong and Han, 2004) use multiple base learners to form an ensemble learner (model) to improve generalization and model performance.

The models can be generated using the same ML algorithm or a combination of algorithms. Ensemble methods are commonly used on imbalanced datasets with data over-sampling techniques.

The ensemble prediction is obtained from a suitable aggregation rule: plurality or majority voting, (weighted) predictions' mean, best performance model, or using learning systems to combine predictions (Zhou et al., 2002).

Through the use of multiple models, ensemble deep learning can improve generalization as well as prediction performance (Yang et al., 2021). Using base models generated from fine-tuning pre-trained language models is expected to further improve performance due to the large context and semantically aware base models.

Ensemble methods provide an improved performance based on the diversity of multiple models. Creating a diverse committee of base learners that

is still consistent with the training data was demonstrated to be of high importance in generating a good ensemble (Dietterich, 2000) as predictions from each learner are combined into the classification outcome.

Methods to generate diversity such as bagging (Breiman, 1996) and boosting are usually applied. In bagging ensembles, multiple models are created using data subsampling methods (i.e. random drawing with replacement) and a joint prediction is obtained through a voting mechanism. In boosting using AdaBoost (Freund et al., 1996) and its variants, data is sampled according to weights assigned to instances and sampling weights are updated based on classification outcomes to improve the scores. These meta-learner methods can be applied to any base learner, including deep learners.

Using the disagreement of an ensemble member with the ensemble's prediction as a diversity measure, (Melville and Mooney, 2004) show that there is a significant Spearman rank correlation between diversity and error reduction of the ensemble. They conclude that increasing ensemble diversity leads to reducing generalization error of the ensemble.

The success of these meta-learners has led to applications in a variety of fields, including text categorization (Shapire and Singer, 2000; Dong and Han, 2004) also revealing some weaknesses. While bagging can reduce the error due to variance of the base classifier, using stable learners, such as Naive Bayes, will not reduce the error. Also, small datasets can generate a limited amount of diversity.

Boosting can perform poorly with insufficient data (Freund and Schapire, 1999) or noisy labels (incorrect class labels in training) (Dietterich, 2000). Other drawbacks for deep model ensembles are the high costs of computing power, memory and process time when training multiple deep learners (Yang et al., 2021).

Using transfer learning from pre-trained transformer language models and the available cloud computing GPUs, our work investigates the feasibility of deep learning ensemble models for text classification. We use a bagging-type data resampling with homogeneous and heterogeneous ensembles to assess how data diversity and type of pre-trained model improve the ensemble model.

4 Legal Datasets

In recent years, curated collections of legal texts have been made available, along with their dedicated language models as: CUAD (Hendrycks et al., 2021), ECHR (Chalkidis et al., 2019), EURLEX57k (Fergadiotis et al., 2018). Based on these, several benchmark datasets included in LexGLUE (available on the HuggingFace[1] platform) (Chalkidis et al., 2021) enable comparison of various AI approaches using the same datasets and metrics.

The LexGLUE multi-labelled datasets (ECtHR, EUR-LEX) include the ECHR-A and ECHR-B datasets containing case descriptions of the European Court of Human Rights, where labels represent articles of the European Convention on Human Rights that have been violated or allegedly violated. LexGLUE also includes EUR-Lex data , which consists of EU legislation that has been labelled according to 7,000 EuroVoc concepts[2] (with 4 granularity levels). The available EUR-Lex data in LexGLUE contains 65,000 documents annotated with the 100 most frequent concepts from level 2.

LexGLUE datasets are designed for Natural Language Undersanding (NLU) and include a temporal 'concept drift' in the datasets for development and test (i.e. data issued at a later time, within five years of training set documents). Since in our study we do not approach NLU problems, we use only the original training datasets which we split into new, smaller train/test sets that both include data collected within the same time interval, limiting the 'concept drift'.

Here we only use the training data of the multi-labelled datasets from the LexGLUE set (ECHR-A, ECHR-B and EUR-Lex datasets) and sub-sample from each, the datasets A, B and C (respectively) without the 'concept drift'. The datasets A and B were sub-sampled using the custom function described in Section 5.1 while for dataset C we select from the EUR-Lex train data only the instances labelled as 'Regulations' and take 50 of the most frequent labels (out of 2941, see Table 1). The unlabelled instances (where these are included in the original train set) have also been removed.

The complexity characteristics of the new datasets are shown in Table 1 in comparison to the original data, where the labels' Cardinality (*Car*) and Density (*Den*) for a dataset with N the number of instances, Y_i the set of labels for instance i and L the number of instances are defined as:

[1] https://huggingface.co/datasets/lex_glue

[2] https://eurovoc.europa.eu

Dataset	N	L	Car	Den	IR
ECHR-A train	8086	10	1.32	0.39	114.00
ECHR-B train	8866	10	1.48	0.45	67.12
EUR-Lex train (Regulations)	29600	2941	4.94	0.002	3554.00
A	5651	10	1.23	0.37	125.34
B	5925	10	1.35	0.41	89.98
C	7201	50	3.34	0.07	8.77

Table 1: Data complexity for the sub-sampled datasets

$Car = \frac{1}{N}\sum_{i=1}^{N}|Y_i|$; and $Den = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i|}{L}$; while the maximum imbalance ratio (IR) is the ratio of the most common label against the rarest one. The new datasets have a smaller number of instances (close to that of custom datasets) and an IR that is higher for datasets A and B and much lower for dataset C.

4.1 Dealing with large text lengths

One characteristic of text in the law and regulatory domains is the large text length, often exceeding 5000 words/entry. Using pre-trained transformers with low to moderate computational demands (as BERT, ALBERT, RoBERTa) is a cost-effective approach to perform MLC on legal text, with limitations due to their maximum input sequence length.

General-purpose NLP transformers can process a maximum input length of 512 tokens, much smaller than the text length in legal datasets. This problem is usually solved by applying 'text truncation', 'hierarchical methods' or 'data transformation'. Text truncation uses only the 'head', 'tail' or 'head + tail' parts of the document, considered to contain the key information. In 'hierarchical' methods, the text is split into sequences (of lengths smaller than 510) their pre-trained embedded representations are extracted from the [CLS] token in the last hidden layer and then combined using max-pooling or mean-pooling (attention weighted) to obtain the embeddings representation of the entire text for classification (Sun et al., 2019).

Other hierarchical methods use hierarchical transformers that generate embeddings and encode these again using a shallow transformer (Chalkidis et al., 2021). The 'data transformation' method (applied here) performs text splitting into sections of suitable length using them to create an expanded dataset (preserving the associated labels). For inference, these entries can be re-joined after fine-tuning along with their predictions.

While truncation methods can miss important information, the 'hierarchical' methods (which create embedding representations of the whole input text) and the 'data transformation' method that re-joins text sections and predictions can both improve prediction outcomes.

5 Methods

5.1 Generating diverse datasets

Real-life, custom legal datasets have a small number of instances and are highly imbalanced – requiring efficient data sampling/splitting methods that enable all labels to be represented in both train and test sets. The iterative method for stratified sampling of imbalanced multi-label datasets proposed by Sechidis et al. (2011) requires reaching convergence, which is time-consuming when used for repeated sub-sampling.

Here we have designed a fast, pseudo-stratified sampling function that provides a train/test split configured to follow the label distribution in the original data and ensuring that all labels are represented in both train and test sets.

C: corpus of labelled text instances
L: set of all labels
M: desired minimum instances per label (M=10)
ratio_in: proportionality factor (configurable)
N_l: number of instances to be selected for label l
labelled(i,l): true iff instance i is labelled with label l
distinct_labels(I): set of all distinct labels associated with instances in set I
choose_random(I,n): set of n randomly chosen instances from set I

for $l \in L$ **do**
  $I_l = \{i | i \in C \wedge \text{labelled}(i,l)\}$
  if $|I_l| < M$ **then**
    $N_l = 1$
  else
    $N_l = |I_l|/\text{ratio\_in}$
  end if
end for
do
  $TEST = \emptyset$
  for $l \in L$ **do**
    $TEST = TEST \cup \text{choose\_random}(I_l, N_l)$
  end for
  $TRAIN = C \setminus TEST$
while distinct_labels$(TEST) \neq$ distinct_labels$(TRAIN)$

Figure 1: Pseudo-code describing the basic algorithm for the pseudo-stratified sampling function

This pseudo-stratified sampling method is designed to randomly extract (without replacement) instances that represent each label in a number proportional to the frequency of each label. We define two integer parameters: M (related to the minimum number of instances per label) and 'ratio_in', which configure the number of entries to be extracted for the test set for each label (basic algorithm described in Figure 1).

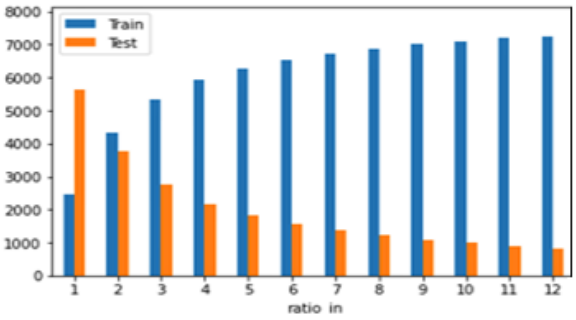


Figure 2: Train and Test size after split using the proposed pseudo-stratified function, for a range of 'ratio_in' values (starting from the ECtHR-A train dataset).

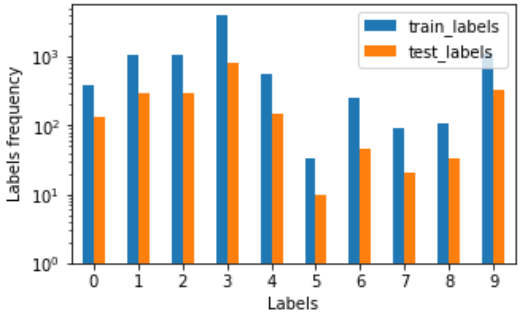


Figure 3: Labels' distribution when splitting the ECtHR-A training dataset into train/test using 'ratio_in'=7

For the training set we keep only the instances that are not included in the test set. The 'ratio_in' values can be chosen from a value of 2 up to a maximum depending on availability of instances for each label. A plot of the train and test sizes for various ratios is shown in Figure 2.

This method provides an efficient splitting ensuring that all labels are represented in both train and test sets. Low values of the 'ratio_in' (2 to 6) give a close representation of the original label distribution, while values above 10 generate only a roughly similar distribution. It can be argued that more data diversity can be obtained using a more distorted distribution. The typical label distribution for the train and test sets is shown in Figure 3.

5.2 Data preparation

The main data is one-hot-encoded and split as train/test datasets, then the train set is further split by re-sampling into train/test sets to create the base-models.

We apply the 'data transformation' method, splitting each text entry into sections of up to 120 words (such that only entire sentences are included in each sequence) while preserving the corresponding labels, creating an expanded dataset. The split sequences are then re-joined along with their predictions at testing or inference time.

The medium and average lengths of the sequences in each dataset are shown in Table 2. Other improvements of the sequence content (i.e. adding to each sequence the last sentence from the previous sequence) are not applied here.

Datasets	A		B		C	
	train	test	train	test	train	test
Median length	108	108	103	105	73	64
Mean Length	104.29	104.42	94.84	96.11	72.51	69.5

Table 2: The medium and mean lengths of sequences in the tran and test sets for the datasets A, B and C after data transformation

The 'data transformation' method is context and semantic aware within each sequence, as each is expected to contain a representative amount of information, that enables classification. Moreover, such phrases generally refer to a specific topic – identified by a certain label group. The 'data transformation' generates much larger train and test datasets on which the model is defined, being a type of boosting technique.

An advantage of text splitting into sequences is that we obtain an increased number of instances, improving the statistical basis of our model (and its bias). This is a type of 'data over-sampling' leading to an ensemble model where predictions are obtained by aggregating over each original instance, which improves prediction variance.

5.3 Generating deep learning ensembles

The proposed method for generating diverse datasets is applied for supervised multi-label classification of legal datasets, creating a train/test split for each. The text for every entry in the train and test datasets is split into sequences of up to 120 words (as described in Section 5.2) forming an expanded dataset where each sequence keeps its original labels, on which each base-model is defined. For testing after fine-tuning, these 'split' entries are re-joined along with their predictions generating an aggregated prediction set.

We create homogeneous and heterogeneous ensembles of deep models, where base-models are fine-tuned from the following pre-trained transformer models: BERT-based-uncased, RoBERTa-base, ALBERT-base-v2 and XLNet-base-cased. The **simpletransformers**[3] library (based on PyTorch) has been used, taking advantage of the unified data, model output formats and default model arguments available (i.e. batch sizes of 16, sequence length of 128, learning rate of 4.e-5).

[3] https://simpletransformers.ai/

To enable generalization, the base learners can be 'incomplete' (not fully trained) models and as the fine-tuning reaches convergence fast (within 3 epochs), we have used only 2 epochs for fine-tuning the base-learners when working on datasets sub-sampled from ECtHR-A and ECtHR-B (10 labels) and on 5 epochs for datasets sub-sampled from the EUR-Lex dataset, (from which we selected data including the most frequent 50 labels). Label weights are applied during training of the model as a means of regularization. Label weights are proportional to labels' frequency in each newly sub-sampled dataset for each base model. The Label Ranking Average Precision[4] (LRAP) metric is used for fine-tuning with the same threshold of 0.5 for all labels.

A set of 10 base-models has been generated for each homogeneous ensemble, each base-model being fine-tuned on a newly resampled dataset. The resampling is performed using the described pseudo-stratified random sampling function, which is used as a type of bagging technique (sampling without replacement to generate the train and test data). As the fine-tuning for the 10 base-models is performed within a loop, to ensure independent results, the model outputs and checkpoints are deleted at the end of each fine-tune and the pre-trained model type is re-initialized before starting the fine-tuning for the next base model.

The base-model prediction arrays are obtained from each 'raw_output' (as given by the prediction outputs of the **simpletransformers** library) after applying a *sigmoid* activation. We aggregate the base-models as the mean of their prediction arrays thus generating the ensemble model (which is also a prediction array).

In the present study we use three types of optimization: (i) we fine-tune (optimize) each classifier using LRAP metrics (the average over each ground truth label assigned to each instance, of the ratio of true vs. total labels with lower score); (ii) we use micro-F1 with thresholding to optimize the predictions on the expanded dataset and find the best three models; (iii) to find the optimal prediction threshold on the re-joined dataset, we use sample averaged F1 as the harmonic mean of Precision (P) and Recall (R) averaged over the sample size S where Y_i and $h(x_i)$ are the true and the predicted

[4] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.label_ranking_average_precision_score.html

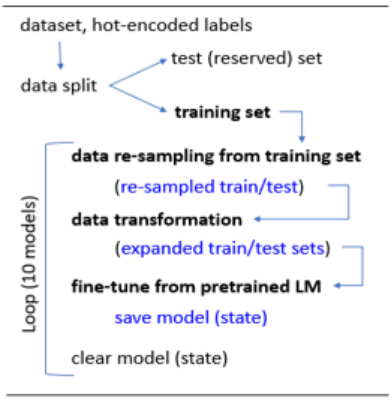


Figure 4: Schematic of the workflow generating the deep ensemble model.

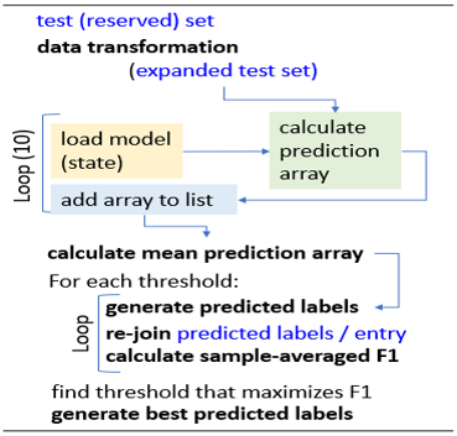


Figure 5: Schematic of the workflow for test and inference using the deep ensemble model.

labels for an example x_i: P = $\frac{1}{S}\sum_{i=1}^{S}\frac{Y_i \cap h(x_i)}{h(x_i)}$;

R = $\frac{1}{S}\sum_{i=1}^{S}\frac{Y_i \cap h(x_i)}{Y_i}$; F1 = $2\frac{PR}{P+R}$

The choice for the optimizations (ii) and (iii) was made based on best results obtained using these types of performance measures for the expanded and re-joined datasets, respectively.

5.4 The workflows

The workflow for generating the ensemble model of deep base-learners is shown in Figure 4 while the workflow for testing and inference is shown in Figure 5 . Both show a concise, intuitive overview of the main steps in generating the models and performing inference.

For inference, we load the saved models, generate the prediction array from each base model and their ensemble as the mean prediction array. For a given threshold in the range 0.1 - 0.9 we generate the predicted labels for the expanded dataset, then aggregate (re-join) this back into the original, while keeping all unique labels obtained from each split text entry. The threshold optimization is per-

formed by seeking the threshold that gives the F1 maximum (calculated as sample averaged).

6 Experiments

The 'main' datasets (A, B and C) have been subsampled from each original dataset (ECtHR-A, ECtHR-B and EUR-Lex, respectively) using the pseudo-stratified sampling function at ratio_in=7, extracting a 'main' dataset (Table 1) and a test dataset (for inference) with a size of about 25-30% of the 'main' dataset.

During fine-tuning, the 'main' dataset is split into train/test using the pseudo-stratified sampling function at ratio_in =5 to generate diverse models. The train/test datasets are pre-processed (according to the description in Section 5.2) obtaining expanded datasets with a maximum text length of 120 words per entry. Fine-tuning is performed on the expanded datasets optimizing the LRAP metrics using model configuration arguments defaults from the **simpletransformers** library.

Several types of deep ensembles are generated: (a) homogeneous and (b) heterogeneous, where the ensemble is created from base-models fine-tuned on the same pre-trained model, or on different ones, respectively. For each (a), and (b) two types of ensemble predictions are then constructed: the 'mean model' by averaging the prediction arrays of the 10 base-learners and the 'best three' model by averaging the prediction arrays of the best three models.

The best three models are selected as those reaching the highest micro-F1 values at the threshold that maximizes micro-F1 on the expanded dataset, obtaining an optimal threshold at values of 0.2-0.3 for the A and B datasets and of 0.5 for dataset C.

The ensemble prediction arrays are optimized using thresholding applied on the 're-joined' dataset with merged predictions, choosing as threshold the value that maximizes the sample averaged F1. For the A and B datasets the optimal threshold is 0.8-0.85 while for dataset C the optimal threshold is 0.5. All predictions with probabilities above the set threshold are kept for each instance.

7 Results and Discussion

Homogeneous ensemble models have been created using the four pre-trained models considered. The ensembles are aggregated as the mean prediction over 10 models, the mean prediction over the best three models, and compared to the best model. The best models are those with the highest micro-F1 values after applying thresholding.

	Expanded data			**Re-joined data**		
	Mean	**Best3**	**Best**	**Mean**	**Best3**	**Best**
RoBERTa	0.73	0.73	0.72	**0.81**	0.8	0.78
XLNet	0.73	0.73	0.71	**0.81**	**0.81**	0.78
BERT	0.73	0.66	0.73	**0.81**	**0.81**	0.78
ALBERT	0.72	0.73	0.72	**0.81**	0.78	**0.81**

Table 3: Optimised micro-F1 scores for dataset A (expanded and re-joined data) on homogeneous ensembles.

	Expanded data			**Re-joined data**		
	Mean	**Best3**	**Best**	**Mean**	**Best3**	**Best**
RoBERTa	0.71	0.71	0.69	**0.79**	**0.79**	0.77
XLNet	0.71	0.7	0.69	**0.79**	**0.79**	0.76
BERT	0.71	0.7	0.68	0.78	0.78	0.75
ALBERT	0.7	0.69	0.68	0.78	0.78	0.76

Table 4: Optimised micro-F1 scores for dataset B (expanded re-joined data) on homogeneous ensembles.

	Expanded data			**Re-joined data**		
	Mean	**Best3**	**Best**	**Mean**	**Best3**	**Best**
ROBERTA	0.74	0.74	0.73	0.79	0.78	0.78
XLNET	0.75	0.75	0.74	**0.80**	0.79	0.78
BERT	0.74	0.73	0.72	0.78	0.78	0.77
ALBERT	0.73	0.74	0.73	0.77	**0.80**	0.76

Table 5: Optimised micro-F1 scores for dataset C (expanded and re-joined data) on homogeneous ensembles.

	Expanded dataset			**Re-joined dataset**		
F1 scores	**micro**	**macro**	**weigh.**	**micro**	**macro**	**weigh.**
dataset A	0.74	0.67	0.73	0.82	0.75	0.82
dataset B	0.73	0.65	0.68	0.79	0.67	0.80
dataset C	0.75	0.67	0.74	0.79	0.71	0.81

Table 6: Optimized F1 scores for the heterogeneous ensemble model using the Best 3 models from each model type on datasets A, B and C.

Comparative results for threshold optimized micro-F1 for the ensemble models calculated as the 'Mean' (over the 10 prediction arrays), 'Best 3' (mean of best three models' predictions) and 'best' (predictions from the best model) are shown in Table 3, Table 4 and Table 5 for datasets A, B and C, respectively. Table 6 shows F1 scores for the heterogeneous ensembles built on the four types of fine-tuned models using the mean prediction from each 'Best 3' models).

While scores' improvements between the homogeneous models and heterogeneous ones are only within 1-2%, there is an important increase of about 5-8% in scores between the expanded model (based on the sequence split dataset) where average optimized micro-F1 is 0.73 and the one for the re-joined data with micro-F1 averages at 0.81 (i.e:

	LinSVC			MultinomialNB		
	micro	**macro**	**weigh.**	**micro**	**macro**	**weigh.**
dataset A	0.8	0.69	0.79	0.47	0.09	0.33
dataset B	0.76	0.64	0.75	0.44	0.09	0.29
dataset C	0.85	0.71	0.83	0.58	0.33	0.44

Table 7: Baseline scores for micro, macro and weighted averaged F1 from sklearn LinSVC and MultinomialNB multi-output methods.

values from Table 3).

The expanded dataset created from sequences with their original labels generates another type of ensemble (a type of over-sampling) which is aggregated at prediction time by joining predictions belonging to each original entry. This type of ensemble appears to be more efficient than aggregating (as a mean) prediction arrays.

In Table 7 we show results obtained for the same datasets, using a strong baseline, the **tf-idf** text processing with LinSVC linear support vector classification from **sklearn** multi-label classification. The LinSVC baseline has high scores, especially for dataset C, as the LinSVC algorithm with the one-vs-rest approach performs very well for datasets with a low imbalance ratio (Table 1). Other multi-output classifiers as 'Multinomial Naive Bayes' cannot handle the data complexity and obtains low scores.

8 Conclusions

We performed multi-label classification on imbalanced datasets sub-sampled from legal text datasets provided in LexGLUE, using deep learning ensembles of base-learners built as fine-tuned transfer models on four well-known NLP transformer models (BERT, RoBERTa, ALBERT and XLNet).

We designed a pseudo-stratified sampling method for imbalanced multi-label datasets to resample diverse datasets which were used to generate base-models for homogeneous deep-ensembles.

Using the GPUs on Google-Colab the training times per epoch for the base models were in the range of 2 - 15 min/epoch. Training times increase with the number of instances as do the inference times (between 1 min - 3 min per model), leading to acceptable times for generating and using such ensemble models.

The values for micro-F1 scores from homogeneous and heterogenous ensembles on the expanded datasets reach close values that improve very little from the 'best model' score (only by 1% - 2%). This also occurs for the scores on the re-joined datasets. Such small differences in the overall outcomes show the close performance of the fine-tuned models, even when starting from different transformer models. These results were also due to the threshold optimization applied, which levels out the original difference in outcomes.

Nevertheless, there is an important score improvement between the expanded and re-joined datasets, indicating that the ensemble generated by text splitting can substantially improve classification outcomes at aggregation time, as threshold optimized scores for the re-joined dataset are 7%-10% higher than those obtained on the expanded dataset.

Overall, the best micro-F1 scores obtained on the re-joined dataset reach 0.78-0.82 which are satisfactory results for imbalanced multi-label datasets of up to 50 labels. Higher scores can be obtained by fine-tuning with optimized model hyperparameters to obtain improved base-models.

The text length in each entry and the degree of label imbalance play a significant role in the fine-tuning performance. For the deep learning models, dataset C (50 labels) with short mean text length per entry and low imbalance obtained similar scores to those for datasets A and B (10 labels) with larger text length/entry and high imbalance. The LinSVC baseline (using one-vs-rest scheme) obtained lower scores on datasets A and B than on dataset C, due to difficulties of binary-relevance type algorithms on highly imbalanced data.

As we have considered small datasets, these can only generate a limited amount of diversity, leading to less variability in the ensemble models. To improve results using ensembles, improved methods to create diversity in datasets should be tested.

Acknowledgements

This work is supported by the Department of Enterprise, Trade and Employment of the Government of Ireland, and by Enterprise Ireland, through the Disruptive Technologies Innovation Fund (Grant No. DTIF2019-074 "TRANSPIRE").

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *arXiv:2004.05150*.

Jasmin Bogatinovski, Ljupčo Todorovski, Sašo Džeroski, and Dragi Kocev. 2022. Comprehensive comparative study of multi-label classification methods. *Expert Systems with Applications*, 203:117215.

Leo Breiman. 1996. Bagging Predictors. *Machine learning*, 24.2:123–140.

Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. 2019. Neural Legal Judgment Prediction in English. *arXiv:1906.02059v1*.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. Legal-bert: The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael James Bommarito, Ion Androutsopoulos, Daniel Martin Katz, and Nikolaos Aletras. 2021. LexGLUE: A Benchmark Dataset for Legal Language Understanding in English. *SSRN Electronic Journal*.

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

Thomas G Dietterich. 2000. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems. MCS 2000.*, volume 1857, pages 1–15.

Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. CogLTX: Applying BERT to long texts. *Advances in Neural Information Processing Systems*, 2020-December(NeurIPS).

Yan-Shi Dong and Ke-Song Han. 2004. A comparison of several ensemble methods for text categorization. In *Proceedings of the 2004 IEEE International Conference on Services Computing (SCC'04)*, pages 0–3. IEEE.

Manos Fergadiotis, Prodromos Malakasiotis, Ion Androutsopoulos, and Ilias Chalkidis. 2018. Large-Scale Multi-Label Text Classification on EU Legislation. *arXiv:1906.02192v1*.

Yoav Freund and Robert E Schapire. 1999. A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780.

Yoav Freund, Robert E Schapire, and Murray Hill. 1996. Experiments with a New Boosting Algorithm. *icml*, 96:148–156.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *arXiv:2006.03654*.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated nlp dataset for legal contract review. *arXiv preprint arXiv:2103.06268*.

Nikita Kitaev, Anselm Levskaya, and Łukasz Kaiser. 2020. REFORMER: THE EFFICIENT TRANSFORMER. In *ICLR 2020*, pages 1–12.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, and Sanjana Mendu. 2019. Text Classification Algorithms : A Survey. *information*, 10(150):1–68.

Zhenzhon Lan, Chen Mingda, Goodman Sebastian, Gimpel Kevin, Piyush Sharma, and Soricut Radu. 2020. ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS. In *ICLR 2020*, pages 1–17.

D Lewis. 1995. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 246–254.

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*.

Gjorgji Madjarov, Dragi Kocev, and Dejan Gjorgjevikj. 2012. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45:3084–3104.

Prem Melville and Raymond J. Mooney. 2004. Diverse Ensembles for Active Learning. In *Proceedings of the 21st International Conference on Machine Learning, (ICML-2004)*, pages 584–591, Banff, Canada.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and New Zealand. 2008. Multi-label Classification using Ensembles of Pruned Sets. In *Eighth IEEE International Conference on Data Mining*, pages 995–1000.

Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2011. On the stratification of multi-label data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6913 LNAI(PART 3):145–158.

Robert E. Shapire and Yoram Singer. 2000. BoosTexter : A Boosting-based System for Text Categorization. *Machine Learning*, 39:135–168.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to Fine-Tune BERT for Text Classification? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11856 LNAI(2):194–206.

Piotr Szyma. 2019. scikit-multilearn : A scikit-based Python environment for performing multi-label classification. *Journal of Machine Learning Research*, 20:1–22.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020a. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020b. Efficient Transformers: A Survey. *ACM Computing Surveys*, pages 1–28.

Grigorios Tsoumakas and Ioannis Vlahavas. 2007. Random k -Labelsets : An Ensemble Method for Multilabel Classification. In *European conference on machine learning. Springer, Berlin, Heidelberg*, pages 406–417.

Yongquan Yang, Haijun Lv, and Ning Chen. 2021. A Survey on Ensemble Learning under the Era of Deep Learning. *arXiv:2101.08387*.

Zhilin Yang, Zihang Dai, Yiming Yang, and Jaime Carbonell. 2019. XLNet : Generalized Autoregressive Pretraining for Language Understanding. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, NeurIPS, pages 1–11, Vancouver, Canada.

Nan Ye, Kian Ming A. Chai, Wee Sun Lee, and Hai Leong Chieu. 2012. Optimizing F-Measures: A Tale of Two Approaches. In *Proceedings of the 29th International Confer- ence on Machine Learning, Edinburgh, Scotland, UK*. arXiv:1206.4625.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 2020-December(NeurIPS).

Zhi-hua Zhou, Jianxin Wu, and Wei Tang. 2002. Ensembling neural networks: Many could be better than all . *Artificial Intelligence*, 174(18):1570.

Handling Class Imbalance when Detecting Dataset Mentions with Pre-trained Language Models

Yousef Younes, Brigitte Mathiak
GESIS – Leibniz-Institute for the Social Sciences
Cologne, Germany
{yousef.younes,brigitte.mathiak}@gesis.org

Abstract

Understanding the links between datasets and publications is a hard task. Yet it is very important to improve dataset discovery and FAIRness of research data. However, only a few datasets with such high-quality links exist. Human annotations are the gold standard for producing such links, but it is a time-consuming manual task, much of which is spent reading text that is not connected to dataset mentions at all. In this paper, we propose a filter to pre-screen scientific publications' sections, so that we can find candidates for dataset mentions reliably. For this, we test both BERT and RoBERTa on sections content as well as on sections title. The main challenge is the inherent imbalance in the data, which we tackle using different imbalance handling techniques, such as re-sampling and variations of the loss function. The best result was obtained when using RoBERTa on section contents by combining re-sampling, balanced focal loss, and a recall-biased validation metric to get a fairly high recall and acceptable precision. The source code and the best obtained model are available here [1].

1 Introduction

Manual textual annotation is a very cumbersome and expensive process, yet it acts as the gold standards that is used for training computational learning models (Pustejovsky and Stubbs, 2012). Unfortunately, this importance turned to be a bottleneck for NLP because the annotation is a time-consuming, laborious, yet error-prone process. To alleviate these difficulties, a lot of tools have been introduced to pre-process the data in order to reduce the amount of human effort needed and to make that effort more focused on the essential task that can not be automated (Pan and Yang, 2009). Detecting research dataset mentions in text (Fan et al., 2022) is an example of an NLP task that needs manually annotated data to train a model.

Research datasets play a crucial role in science. They act as a unifying point that allows comparison of different research ideas and also facilitate reproducibility of results (Wilkinson et al., 2016). For these reasons, the problem of finding datasets mention in research papers has gotten more attention recently (Zhao et al., 2018). To tackle this problem, researchers like in (Färber et al., 2021)(Mesbah et al., 2018) treat it as a domain-specific, supervised NER task that needs annotated dataset to operate on. The Coleridge dataset [2] used in Kaggle's "Show US the Data" competition is an example of such dataset in which research papers are annotated with the datasets mentioned in them. One of the challenges that annotators face when doing such annotation is the sparsity of these mentions in scientific text. By sparsity, we mean that most of the texts do not have dataset mentions and only a small amount contains such mentions. This sparsity has negative consequences not only on the annotation process but also on the annotation result. During annotation, the annotators spend a lot of time reading texts that do not contain dataset mentions. Naturally, the resulting annotated dataset is imbalanced towards the samples without mentions.

One way to help annotators is to perform blocking (Azzalini et al., 2021) in order to pick the texts that contain dataset mentions so they can focus on them. To achieve this, we define a binary classification task in which we classify text as belonging to the negative class (N) when it does not have dataset mention and to the positive class (P) when it has. For example, the following excerpt "...classification problem on a subset of *ADNI database* consisting

[1] https://github.com/YousefYounes15/dataset_mention_detection

[2] https://www.kaggle.com/c/coleridgeinitiative-show-us-the-data/data

of 531 subjects with sMRI and DTI scans..." makes the paper's section that contains it belongs to the P class since ANDI is an abbreviation for a dataset. From here onward, N and P will be used to refer to the negative and positive classes respectively.

Our goal is to get a binary classifier with the highest possible P-recall and high precision as well. Such classifier will help reduce the time and effort needed for the manual annotation. To achieve this goal we chose to work with language models assuming that the context plays a vital role for the task at hand. Since our data is imbalanced, we used the chance to compare the performance of two popular models: BERT (Kenton and Toutanova, 2019) and RoBERTa (Liu et al., 2019) on such data. To the best of our knowledge, there is no such comparison in the literature. We have experimented on different parts of sections content and also on the sections title to find out which settings work better. We also have used different imbalance remedy techniques like re-sampling and cost-sensitive learning i.e., balanced focal loss.

The contributions of this paper can be outlined in the following points:

- Re-sampling has more effect than cost-sensitive learning on language model-based classifiers.
- We can not depend on sections title to perform this classification using language models, albeit dataset mentions are more present under particular section titles.
- RoBERTa outperforms BERT even when the data is imbalanced.
- The best results with a P-recall of 86% were achieved with RoBERTa when combining re-sampling, cost-sensitive learning, and a recall-oriented validation metric.

The rest of this paper is organized as follows. Section 2 reviews how text classification was improved by word embeddings before it summarizes bias handling techniques. Section 3 describes the data that we used in the paper before it gives details on the different loss functions used in the experiments. Section 4 describes the experimental settings then it reports the results of the different experiments. After that, the results are discussed in section 5. Finally, the paper concludes in section 6.

2 Related Work

2.1 Text Classification with Deep Learning

Text classification is a supervised machine learning task in which a given piece of text is predicted to belong to a class of a set of predefined classes (Vijayan et al., 2017). To perform this task, many traditional algorithms such as Naive Bayes, Support Vector Machines (SVM), Tree-based models (Kowsari et al., 2019); and modern neural-based algorithms such as RNN, CNN, DBN have been used (Minaee et al., 2021). All of these models share a common problem to which each has devised a different solution which is text representation.

Transfer learning is a learning framework that revolutionized the field of machine learning by breaking the isolation of both tasks and models (Pan and Yang, 2009). It enables a model which is trained on one task to transfer the knowledge it gained to another related task via fine-tuning (Torrey and Shavlik, 2010). As a sub-field of machine learning, NLP witnessed a huge leap due to transfer learning thanks to the introduction of pre-trained word embeddings.

Word embeddings are one of the most successful examples of transfer learning because they are learned in one task and used to solve other tasks. Earlier word embeddings such as word2vec (Mikolov et al., 2013a)(Mikolov et al., 2013b) and GloVe (Pennington et al., 2014) represent a word by a high-dimensional vector. The similarities between words are reflected as similarities between their vector representations. These embeddings are usually used to initialize new models but the whole model needs to be re-trained from scratch on the new data. That is why they are known as shallow transfer learning. Contextual embeddings are an improvement over word embeddings in which the context of the word is involved in its representation (Liu et al., 2020). Pretrained language models such as ELMo (Peters et al., 2018), GPT-n (Radford et al., 2019)(Brown et al., 2020), BERT (Kenton and Toutanova, 2019), RoBERTa (Liu et al., 2019) are used to obtain this type of representation. Unlike word embeddings, these models are usually used as part of new models which are fine-tuned on the data for a particular task. This significantly reduces the need for huge training data thus reducing the computational cost.

The great success achieved by language models clearly indicates the vital role played by context in natural language processing. Dataset mentions in

the scientific text are not an exception. Although there is no standard method to mention a dataset in a research paper, we assume that there is some kind of unwritten rules that govern the way researchers mention a dataset. These rules have something to do with the context in which the dataset title or label is mentioned. For this reason, we make use of language models to solve our problem. More particularly, the focus is on BERT and RoBERTa since they provide a contextual embedding that proved to be successful for many NLP tasks. In addition, we take the chance to compare the performance of these two models on imbalanced data.

The classifiers used in the experiments are constructed by adding one linear layer on top of the base version of BERT and RoBERTa. So in the rest of this paper, we will be using BERT and RoBERTa as shortcuts to indicate the classifiers based on them. While this is a very straightforward way to turn these into classifiers, due to the inherent limitation of the language model, they can only handle a limited input of up to 512 tokens at a time. One way to address this issue, which was introduced in (Sun et al., 2019), is to select a representative set of tokens from the document that results in the best quality for the classifier. One of our intentions is to use and extend this work for imbalanced data. Another approach is to get a proper document representation by dividing the text into equally-sized chunks then using word-level encoder and pooling followed by chunk-level encoder and pooling as illustrated in (Su et al., 2021).

2.2 Handling Class Imbalance

Class imbalance is a feature that characterizes many labeled datasets. This imbalance has a direct effect on the classifier so many techniques have been proposed to deal with it. These techniques fall under two categories: re-sampling and cost-sensitive learning (Iikura et al., 2020).

Re-sampling methods try to address the problem by duplicating samples of minority classes, removing samples of majority classes or generating new samples of minority classes. These methods have been used successfully for text classification e.g., (Estabrooks et al., 2004)(Tepper et al., 2020). But the removal and duplication of samples change the data distribution and can cause the models to overfit the minority classes. For this reason, approaches like in (Chawla et al., 2002)(Guo and Viktor, 2004) have been introduced to produce synthetic data. In this work, we will use sample removal and duplication and leave sample generation for future work because text generation is out of the scope of this paper.

Cost-sensitive learning methods approach the problem by injecting data bias in the cost function also known as the loss function. The loss function condenses the whole learning system into a single number called the loss whose value must be minimized in order to improve the performance of the model. Cost-sensitive learning tailors the loss in favor of a particular class(es) by multiplying it by a weight value. One way to choose that weight is to use the inverse class frequency like in Balanced Cross Entropy (BCE) (Phan and Yamamoto, 2020). Another way is to use the difficulty of the sample as a weight like in Focal Loss (FL) (Lin et al., 2017).

To the best of our knowledge, a comparison of the effect of imbalance handling techniques on BERT- and RoBERTa-based classifiers was not performed. In this work, we investigate empirically the effect of using re-sampling and cost-sensitive learning on such classifiers.

3 Methodology

This section describes the data and loss functions that are used in this study. It starts by explaining how the data is prepared. Then an overview of the loss functions is presented.

3.1 Data

The experiments in this paper use the Coleridge dataset. This dataset consists of a collection of research papers (∼19.6 K) out of which 14.3K papers are unique and 5.3K papers are duplicated due to the fact that they have multiple dataset mentions. Each of these papers is stored in a JSON-file that wraps each section's title and content in a JSON object. In addition to the JSON files, there is a CSV-file that contains basic metadata (file name, paper title, dataset title, dataset label, cleaned dataset label) about each paper. To prepare the data for our experiments, it was processed as follows: scan through the sections of each paper to extract the title and text. Then a binary label is generated for every section. This label contains (1) if a dataset associated with the paper is mentioned in the section either using its title or its label, otherwise it contains (0). The final dataset that we got contains approximately 233K samples. Each sample consists of the following pieces of information:

(file name, paper name, section title, section text, dataset mention, label). The sections are considered documents whose contents and titles are used in the experiments. Using sections as our basic text units is a compromise between using all of the paper, which would be trivial, and sentence level, which often does not have enough information to make the decision and introduce even more bias. Sections can be easily identified, even when they are not part of the metadata of the publication (Mathiak et al., 2009).

Before we run experiments on this dataset, we divided it randomly into two parts: train and test set. The test set accounts for 20% of the dataset ($\sim$ 47K samples) and is used to test the classifier after training is complete. This testing set is the same for all experiments. The remaining 80% ($\sim$ 187K samples) is kept for training and validation sets. During training, 80% of this data is selected randomly to be used for training and the remaining are used for validation. Since the splitting is done randomly, different sections from the same paper could appear in train, val and test sets. In all operations involving randomization, a seed is used to ensure reproducibility.

Inspecting the data, we have found that among the existing 233297 samples only 27856 samples contain dataset mentions i.e., they belong to the P class. We also have found that the majority of the sections ($\sim$200K) contain long text. In summary, only 12% of the samples in the data belong to the P class. Besides that 85% of the samples contain long text that will represent a challenge to the models used in this work.

3.2 Loss Functions

Focal Loss (FL) was introduced in (Lin et al., 2017) to handle the bias of the object detection problem in computer vision. In this study, we apply it to a binary text classification along with three other functions, which are also variations of cross entropy (Murphy, 2012). They can be generalized in one formula, Eq.1, from which different loss functions can be derived by assigning different values to the parameters.

$$BFL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \qquad (1)$$

Equation 1 can be split into three components. The main component is the binary cross entropy $-log(p_t)$ where p_t holds either the model's estimated probability ($p \in [0,1]$) for class P or its complement $(1-p)$ for class N. Another part is the balancing factor α_t which, when it is activated, takes a different positive non-zero value for each class to reflect its weight. But α_t could be deactivated by giving it the same value for all classes. In the context of this paper, we will write $\alpha_t = (x, y)$ to indicate that x is the weight for class N and y is the weight for class P. Since we have only two classes, we will set $\alpha_t = (1, 1)$ when the class's weight is not considered. Otherwise, α_t will be assigned values that can be chosen by the user or set by some criteria like the inverse class frequency (Phan and Yamamoto, 2020). The third part is the modulating factor $(1-p_t)^{\gamma}$ where $\gamma >= 0$ is the focusing parameter. This modulating factor adjusts the contribution of the sample in the loss based on its difficulty. For difficult samples about which the classifier is not certain, p_t will have low values which will increase the modulating factor resulting in more contribution of these samples in the loss. While for easy samples, p_t will have high values and the modulating factor approaches zero thus reducing the contribution of these samples in the loss. Now let us derive the loss functions.

Cross entropy is used as the loss function in the BERT- and Roberta-based classifiers. It takes neither the class's weight nor the samples' difficulty into account. It can be derived from Eq.1 by setting $\alpha_t = (1, 1)$ and γ= 0. Balanced cross entropy considers the class imbalance but not the difficulty of the samples. To derive it from Eq.1, we set $\gamma = 0$ and α_t could take different values. We will use the classes' percentages in the data $\alpha_t = (0.12, 0.88)$ beside other values in the experiments (cf. section 4.4). Focal loss pays attention to the difficulty of the sample ($\gamma > 0$) but not on the weight of the classes ($\alpha_t = (1, 1)$). Finally Balanced Focal Loss (BFL) brings everything together so $\alpha_i > 0$ for $i = 0, 1$ and $\gamma > 0$.

4 Results

In this section, we start by describing the experimental settings. After that, the results of different experiments are presented.

Our goal is to find the best model for selecting texts with dataset mentions. The major challenges in this are the length of the texts and the inherent imbalance in the data (cf. section 3.1). We start by experimenting with the section contents to find the text features that give the best results. Then we use these features to select the best sampling rates and

loss function settings. After that, we combine all of these results together. Additionally, we experiment with the section titles using different configurations. Finally, we try to improve the results by changing the validation metric.

4.1 Experimental settings

All experiments were implemented in Pytorch and ran on a 4-GPU machine with a GeForce RTX 2080 Ti with 11 GB RAM each. The classifiers were trained for 4 epochs using the AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 2x10-5 to overcome the catastrophic forgetting problem following the recommendation of (Sun et al., 2019). Each epoch consists of one full scan of the training set to compute the loss followed by one pass through the validation set then the accuracy is used as the validation metric to measure the classifier progress. Different loss functions described in section 3.2 will be used. To make full use of GPU memory, the batch size was chosen to be 32 for experiments conducted on section contents and 256 for the ones on section titles (cf. section 4.6). We have made use of the same seed for all operations that involve randomization to make the experiments repeatable. In some experiments, we do diverge from these settings by changing the validation metric which will be clearly stated in the corresponding section.

To make sure that our results are reliable, we adopt the following strategy. We start by running a set of initial experiments to help us find settings that produce the best results. If these settings were in line with results obtained from previous research we use them; otherwise, we run experiments using 5-fold cross-validation to verify our results and use settings that produce the best average result on the test set. In all experiments, we use the test data to report the values of precision, recall and F1-score for both the P and N classes. We also report the accuracy (ACC) and Matthew Correlation Coefficient (MCC) metrics. Since the model is intended to filter samples of the P class, we are mainly interested in improving the P-recall. So we will be comparing models based on their P-recall but in case of a tie, other metrics will be used.

4.2 Feature Selection

In this subsection, we investigate which of the text's features serve our goal the best. Particularly, we are interested in token selection, as we try to keep within the 510 token limit imposed by BERT.

The sections content are used as input for the classifier. In this context, every section is considered a document by itself (cf. section 3.1). BERT accepts a maximum input of 512 tokens (cf. section 2.1), yet most sections are longer than this (cf. section 3.1). We also know that the choice of input tokens for BERT has an impact on the quality of the classifier (Sun et al., 2019). As such, we want to find out which part of the text achieves the best classification results. Working on balanced data, it was found in (Sun et al., 2019) that picking three-quarters of BERT's input tokens from the beginning of the document and one-quarter from the end of the document results in the best classification outcome. Since our data is imbalanced, we want to examine whether that imbalance has an impact on the choice of the tokens. To do so, we run several experiments using different parts of the sections. The results reported in Table 1 show that the tokens (F 382+L 128) produced the best result (P-recall = 0.79 and MCC = 0.835) for this dataset. This complies with the result obtained in (Sun et al., 2019) which implies that the imbalance nature of the data has no impact on the choice of the input tokens for the task at hand. Based on that, we consider BERT with these input settings as our base model. The experiments to follow will use and build upon these settings.

4.3 Handling Imbalance by Under- and Oversampling

As mentioned in the data description, most of the text does not contain dataset mentions and only 12% of the sections belong to the P class. One way to overcome this imbalance is to use re-sampling. Here we will over-sample the P samples, downsample the N ones and combine the two together to find the sampling ratio that gives the best result. Building on the results from the previous experiments, we have used F 382+ L 128 tokens of sections texts as input to BERT. We have run several experiments with different re-sampling settings to find that the best re-sampling factor for the dataset at hand is (4:0.55) which is to quadruple the P samples and take more than half of the N samples. The corresponding approximated ratio for this sampling factor obtained by dividing the number of positive samples by the number of negative samples is (1) indicating that the numbers of P and N samples are roughly the same. Putting it differently, the model works best when the dataset is balanced. To get a

Tokens	Precision		Recall		F1-Score		ACC	MCC
	N	P	N	P	N	P		
F 510	0.97	0.92	0.99	0.78	0.98	0.84	0.97	0.827
L 510	0.97	0.92	0.99	0.74	0.98	0.82	0.96	0.808
F 255 + L 255	0.97	0.92	0.99	0.79	0.98	0.85	0.97	0.834
F 128 + L 382	0.97	0.92	0.99	0.79	0.98	0.85	0.97	0.834
F 382 + L 128	0.98	0.93	0.99	**0.79**	0.98	0.85	0.97	**0.835**

Table 1: Feature Selection for Classifying Section Excerpts with BERT. In the "Tokens" column, F stands for First and L for Last. F 510 means the first 510 tokens. L 510 means the last 510 tokens.

reliable result, we ran an experiment using 5-fold cross-validation using a BERT-based classifier and reported the average testing result in Table 2. Although the accuracy is 95%, the P-recall is only 83%. This is the highest P-recall value that could be obtained using re-sampling as a technique to surpass the imbalance problem.

4.4 Imbalance Handling with Loss Functions

Loss-sensitive learning is another known technique to handle Imbalance. Experiments ran so far used cross entropy to compute the loss. Here we ran several experiments with different loss functions via changing the values of α_t, γ in Eq. 1 as explained in section 3.2. The goal is to find the best loss function settings for our task. To achieve this goal, we experiment not only with BERT but also with RoBERTa. Because, unlike text features and sampling factor which are data-related issues, the loss function is part of the model so we want to compare its effect on the two models.

From the initial experiments in which different values for α_t and γ were utilized, we found that when using the focal loss i.e., (α_t = (1,1) and $\gamma = 2$), RoBERTa outperformed BERT with respect to P-recall. Moreover, with balanced cross entropy ($\alpha_t = (.12, .88)$ and $\gamma = 0$) both models produce similar accuracy and P-recall. The best results for both models were obtained when BFL (α_t = (.12,.88), γ=4) is used, with BFL the P-recall value was 77% and 78% for BERT and RoBERTa respectively. Using higher values for γ i.e., $\gamma > 4$, the P-recall started to decline for both models. We also tried to improve RoBERTa results using different values for α_t but we were not able to get better results. To make sure that the best results that we obtained using BFL (α_t = (.12,.88), γ=4) are stable and not due to some randomness, We ran an experiment using 5-fold cross-validation and reported the average of the results obtained on the testing data in Table 3. These results show that RoBERTa is outperforming BERT with 1% increase in P-recall. The next step is to combine the best settings found so far in one experiment.

4.5 Combining Best Settings

In previous experiments, we have seen the effect of using re-sampling and BFL individually. In this subsection, we will combine all the results that we obtained so far. That is to use 382 tokens from the beginning of the section and 128 tokens from the end as the model's input. Furthermore use the sampling factor (4:0.55) that balanced the dataset i.e., ratio=1. In addition, use balanced focal loss with $\alpha_t = (.12, .88)$ and $\gamma = 4$. We have run an experiment using 5-fold cross-validation for both BERT and RoBERTa with these settings reported the average of all metrics in Table 4. The results for RoBERTa and BERT are competitive in general but RoBERTa achieved 3% increase in P-recall over BERT with these settings.

4.6 Classifying Using Section Titles

Working on the training data, we have noticed that there is a tendency among researchers to mention datasets under sections with specific titles such as Introduction, Methodology, Discussion, Data, Datasets, Results, and Experiments among others. Based on that, we had the hypothesis that we can depend on the section titles to decide whether a section contains dataset mentions or not. To test this hypothesis, we experimented using four configurations (data as is, data with sampling, BFL, BFL with sampling) on both models. The best P-recall (68%) was obtained by RoBERTa when using BFL with sampling. But lower values of other measures such as P-precision = 0.22 and MCC=0.244 indicate that using the titles with these configurations to do the classification task at hand is not promising.

Precision		Recall		F1-Score		ACC	MCC
N	P	N	P	N	P		
0.98	0.80	0.97	**0.83**	0.97	0.81	0.95	0.789

Table 2: Average Testing Results using BERT when Data is Balanced using Re-sampling

Model	Precision		Recall		F1-Score		ACC	MCC
	N	P	N	P	N	P		
BERT	0.97	0.93	0.99	0.76	0.98	0.84	0.97	0.821
RoBERTa	0.97	0.93	0.99	**0.77**	0.98	0.85	0.97	0.830

Table 3: Average Testing Results Using Balanced Focal Loss with $\alpha_t = (.12, .88)$ and $\gamma = 4$

4.7 Experimenting With Validation Metrics

So far the best-known imbalance handling techniques were used individually and combined but we were not able to achieve a P-recall above 83%. Here we will add a new idea: that is to select the validation metric used on the validation set to be in line with our goal. Since our goal is to have a classifier that is able to find the overwhelming majority of the P samples, recall represents an exact match for this goal so we will use it as the validation metric. We ran an experiment using 5-fold cross-validation for each model on both section contents and titles. The average results of the experiments are documented in Table 5. Again, RoBERTa outperformed BERT on sections content with a P-recall of 85%. In other words, using recall as a validation metric resulted in 2% improvement on RoBERTa's P-recall reported in Table 4. Similarly, for the experiment that used the titles, the best P-recall obtained by RoBERTa was 70% which means a 2% improvement compared to the result from section 4.6.

To further improve the results, we have weighted the recall three times more than precision by using F-Beta score as the validation metric. With this, we ran the experiments and reported the results in Table 6. This modification improved the P-recall for RoBERTa on sections content by 1% with 5% decrease on P-precision compared to Table 5. So RoBERTa-based classifier on the sections content with Fbeta is the final result of this paper.

5 Discussion

The re-sampling experiments in section 4.3 show that there are limits for oversampling the P-samples and under-sampling the N-samples after which the performance of the classifier starts to decline. These limits correspond to having the same number of P- and N-samples i.e., achieving perfect balance, which is the expected outcome. The average classifier performance obtained when re-sampling balances the dataset is 83% P-recall and 80% P-precision see Table 2. This is better than the original results (79% P-recall) see table 1, but also encourages us to find even better ways to address the issue.

The experiments with loss functions in section 4.4 show that loss functions have more effect on RoBERTa than on BERT. As a side effect, we find that RoBERTa is better than BERT when dealing with imbalanced data. Nevertheless, using re-sampling with BERT produced better P-recall compared to using BFL with both BERT and RoBERTa. So we can conclude that re-sampling is actually more effective than using BFL in this particular use case. However, as seen in the experiments in section 4.5 combining BFL and re-sampling together improves the overall performance. RoBERTa outperformed BERT with 3% increase on P-recall see Table 4. That means when dealing with balanced data, the impact of the used loss function on RoBERTa is more than on BERT.

Starting from section 4.6, the experiments involved section titles to do the classification task. In these experiments, RoBERTa produced better P-recall than BERT. But the best P-recall value obtained was 70% is 16% lower than P-recall obtained on section contents. This is not surprising due to the lower amount of linguistic knowledge contained in the section titles compared to the section contents. This is in contrast to the findings in (Galke et al., 2017), where they have very promising results with only the titles, but also a very different classification task not related to dataset mentions. In section 4.7, the experimental results show that changing the validation metric has a good impact on the results of both models. When weighing

Model	Precision		Recall		F1-Score		ACC	MCC
	N	P	N	P	N	P		
BERT	0.97	0.81	0.97	0.8	0.97	0.80	0.95	0.779
RoBERTa	0.98	0.75	0.96	**0.83**	0.97	0.78	0.95	0.757

Table 4: Average Testing results using a combination of re-sampling and Balanced Focal Loss.

Setting	Model	Precision		Recall		F1-Score		ACC	MCC
		N	P	N	P	N	P		
BFL	BERT	0.97	0.90	0.99	0.77	0.98	0.83	0.96	0.815
	RoBERTa	0.97	0.91	0.99	0.78	0.98	0.84	0.97	0.820
BFL+S+C	BERT	0.97	0.75	0.96	0.82	0.97	0.78	0.94	0.749
	RoBERTa	0.98	**0.68**	0.94	**0.85**	0.96	0.75	0.93	0.721
BFL+S+T	BERT	0.94	0.22	0.70	0.68	0.80	0.34	0.70	0.252
	RoBERTa	0.94	0.21	0.66	0.70	0.78	0.32	0.66	0.232

Table 5: Average Testing Results Using Re-sampling and Balanced Focal Loss with Recall as Validation Metric. BFL stands for Balanced Focal Loss, C for Content, T for Titles, and S for Sampling.

Setting	Model	Precision		Recall		F1-Score		ACC	MCC
		N	P	N	P	N	P		
BFL+ S + C	BERT	0.98	0.73	0.96	0.81	0.97	0.77	0.94	0.741
	RoBERTa	0.98	**0.63**	0.93	**0.86**	0.95	0.72	0.92	0.692
BFL+ S + T	BERT	0.94	0.23	0.70	0.67	0.81	0.34	0.70	0.255
	RoBERTa	0.94	0.23	0.72	0.63	0.81	0.33	0.71	0.238

Table 6: Average Testing Results Using a Combination of Best Settings with F-beta as Validation Metric

the recall three times more than the precision using the F-beta score, we were able to obtain a classifier with 86% P-recall and 63% P-precision using RoBERTa. In other words, using such a filter will reduce the annotation time by more than 50% at the price of losing 14% of the positive samples.

We went further to investigate why the model was not able to pick the 14% P-samples by looking at some false negative cases which have dataset mentions but the model classified them as not having. We have found that the model has difficulty when the dataset is mentioned indirectly like in "...Empirically, Schwellnus and Arnold (2008) uses data from OECD's firms to show that increases in corporate taxes negatively impact firms...". Here the source of the data is used but the actual name of the dataset is not used. In another case, the distance between the name of the dataset and the word that refers to it was quite long like in "...*Studies* that have collected longitudinal data tend to be based on relatively small samples, whereas the *E CL S - B* provided a large, nationally representative sample a...". Finally, we noticed that the dataset acronyms represent a challenge for the tokenizer.

6 Conclusion and Future Work

The re-sampling experiments in section 4.3 show that the models work best when the data is balanced. The experiments with loss functions in section 4.4 show that, with language models, using re-sampling to handle imbalance is more effective than using loss functions. In general, RoBERTa kept outperforming BERT when using imbalance handling techniques. The set-up with which RoBERTa classifier reached the highest P-recall (86%) was to use a balanced dataset of section contents with a proper loss function and a validation metric that is compatible with the task's goal. Particularly, we used F-beta as the validation metric with recall weighted three times more than precision. We are planning to test this filtering algorithm with real-life annotators next. The idea is to provide them with candidates for extracts from full-text publications that allow them to efficiently find dataset mentions and annotate them. In the long run, we are interested in not only the dataset mentions themselves but also to enrich the mentions with additional metadata, such as direct links to the datasets.

References

Fabio Azzalini, Songle Jin, Marco Renzi, and Letizia Tanca. 2021. Blocking techniques for entity linkage: A semantics-based approach. *Data Science and Engineering*, 6(1):20–38.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, and Amanda Askell. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

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.

Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. 2004. A Multiple Resampling Method for Learning from Imbalanced Data Sets. *Computational Intelligence*, 20(1):18–36.

Lizhou Fan, Sara Lafia, David Bleckley, Elizabeth Moss, Andrea Thomer, and Libby Hemphill. 2022. Librarian-in-the-loop: A natural language processing paradigm for detecting informal mentions of research data in academic literature. *CoRR*, abs/2203.05112.

Michael Färber, Alexander Albers, and Felix Schüber. 2021. Identifying used methods and datasets in scientific publications. In *Proceedings of the Workshop on Scientific Document Understanding: co-located with 35th AAAI Conference on Artificial Inteligence (AAAI 2021) ; Remote, February 9, 2021. Ed.: A. P. B. Veyseh*, volume 2831 of *CEUR Workshop Proceedings*. RWTH Aachen. ISSN: 1613-0073.

Lukas Galke, Florian Mai, Alan Schelten, Dennis Brunsch, and Ansgar Scherp. 2017. Using titles vs. full-text as source for automated semantic document annotation. In *Proceedings of the Knowledge Capture Conference*, pages 1–4.

Hongyu Guo and Herna L. Viktor. 2004. Learning from imbalanced data sets with boosting and data generation: the DataBoost-IM approach. *ACM SIGKDD Explorations Newsletter*, 6(1):30–39.

Riku Iikura, Makoto Okada, and Naoki Mori. 2020. Improving BERT with Focal Loss for Paragraph Segmentation of Novels. In *International Symposium on Distributed Computing and Artificial Intelligence*, pages 21–30. Springer.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information*, 10(4):150. Publisher: Multidisciplinary Digital Publishing Institute.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Qi Liu, Matt J. Kusner, and Phil Blunsom. 2020. A Survey on Contextual Embeddings. *arXiv:2003.07278 [cs]*. ArXiv: 2003.07278.

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. Decoupled Weight Decay Regularization. *arXiv:1711.05101 [cs, math]*. ArXiv: 1711.05101.

Brigitte Mathiak, Andreas Kupfer, and Silke Eckstein. 2009. Using Layout Data for the Analysis of Scientific Literature. In Djamel A. Zighed, Shusaku Tsumoto, Zbigniew W. Ras, and Hakim Hacid, editors, *Mining Complex Data*, volume 165, pages 3–22. Springer Berlin Heidelberg, Berlin, Heidelberg. ISSN: 1860-949X, 1860-9503 Series Title: Studies in Computational Intelligence.

Sepideh Mesbah, Christoph Lofi, Manuel Valle Torre, Alessandro Bozzon, and Geert-Jan Houben. 2018. Tse-ner: An iterative approach for long-tail entity extraction in scientific publications. In *International Semantic Web Conference*, pages 127–143. Springer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: a comprehensive review. *ACM Computing Surveys (CSUR)*, 54(3):1–40.

Kevin P. Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359. Publisher: IEEE.

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.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv:1802.05365 [cs]*. ArXiv: 1802.05365.

Trong Huy Phan and Kazuma Yamamoto. 2020. Resolving Class Imbalance in Object Detection with Weighted Cross Entropy Losses. *arXiv:2006.01413 [cs]*. ArXiv: 2006.01413.

James Pustejovsky and Amber Stubbs. 2012. *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*. " O'Reilly Media, Inc.".

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.

Xin Su, Timothy Miller, Xiyu Ding, Majid Afshar, and Dmitriy Dligach. 2021. Classifying Long Clinical Documents with Pre-trained Transformers. *arXiv:2105.06752 [cs]*. ArXiv: 2105.06752.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Naama Tepper, Esther Goldbraich, Naama Zwerdling, George Kour, Ateret Anaby Tavor, and Boaz Carmeli. 2020. Balancing via Generation for Multi-Class Text Classification Improvement. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1440–1452, Online. Association for Computational Linguistics.

Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global.

Vikas K Vijayan, K. R. Bindu, and Latha Parameswaran. 2017. A comprehensive study of text classification algorithms. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1109–1113.

Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, and Philip E. Bourne. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9. Publisher: Nature Publishing Group.

Mengnan Zhao, Erjia Yan, and Kai Li. 2018. Data set mentions and citations: A content analysis of full-text publications. *Journal of the Association for Information Science and Technology*, 69(1):32–46.

Performance of two French BERT models for French language on verbatim transcripts and online posts

Emmanuelle Kelodjoue and **Jérôme Goulian** and **Didier Schwab**
University of Grenoble Alpes, LIG
Bâtiment IMAG Université Grenoble Alpes
700, avenue centrale 38401 Saint Martin d'Hères
emmanuelle.kelodjoue-nguemegne@univ-grenoble-alpes.fr
jerome.goulian@univ-grenoble-alpes.fr
didier.schwab@univ-grenoble-alpes.fr

Abstract

Pre-trained models based on the Transformer architecture have achieved notable performances in various language processing tasks. This article presents a comparison of two pre-trained versions for French in a three-class classification task. The datasets used are of two types: a set of annotated verbatim transcripts from face-to-face interviews conducted during a market study and a set of online posts extracted from a community platform. Little work has been done in these two areas with transcribed oral corpora and online posts in French.

1 Introduction

Opinion mining has recently undergone a change with the rise of deep learning and, especially, the use pre-trained Language Models (Vaswani et al., 2017). The use of the latter such as ELMO (Peters et al., 2018) and BERT (Devlin et al., 2018) has led to significant improvements on a wide range of NLP tasks for the English language, from relation extraction to document classification (Peng et al., 2019; Laskar et al., 2020). French variants such as FlauBERT (Le et al., 2020) and CamemBERT (Martin et al., 2019) were proposed later on.

In this work, we are interested in the classification of two types of data as being either *in favour* (motivation), either *not in favour* (barriers) or *in favour on the condition that* (condition) :

- Verbatim transcripts from face-to-face interviews conducted in the context of a market potential study of an innovative product using natural language processing methods (NLP).
- Online posts comes from a community platform called *Yoomaneo*.[1]

[1]https://www.yoomaneo.com/

Since we work with French data, we propose to compare and analyse the performance of two pre-trained versions for French. Additionally, since the collected data is small, we propose to augment the data with different augmentation techniques.

Contribution: This paper aims to compare and analyse the performances of two french BERT models on two different types of data.

2 Dataset : Constitution and Annotation

2.1 Dataset origin: Verbatim transcript

The dataset used to build and evaluate the French BERT models in this work comes from a set of 4367 verbatim. These verbatim were manually extracted from 75 transcripts[2] of face-to-face interviews. [3] To use this dataset for our research task, we conduct a human evaluation. We gather 6 evaluators and create two subunits of 3 annotators and add one more[4] to balance the evaluation of the two groups. We ask each group to review monthly 200 verbatim from the 4367.

Evaluation rules:

Only the verbatim whose classification obtained an interrater agreement according to the following rules were kept. Each verbatim of our initial corpus (4367) must be evaluated by at least 3 people. If a class (barrier, motivation or condition) results in an agreement greater than or equal to 50% for a verbatim and there is not a 50/50 on it, the selected verbatim and the assigned class is selected. On the other hand, if the interrater agreement is less than 50% or if there is 50/50 on two labels, the verbatim is eliminated from the corpus. 1578 out of 4367 verbatim transcripts have been evaluated, and only 839 verbatim transcripts obtained an agreement

[2]434 081 tokens.

[3]The interviews were conducted as part of different market potential studies catering various innovative products in the field of electricity, health, electrical goods, gerontology, automatism and pastry.

[4]We called him the *common annotator* since his role is to fill the empty space left by one of the six initials annotators.

greater than or equal to 50%. The distribution of the corpus is given in Table 1.

Classes	Number of verbatim
Barriers	189
Motivation	407
Condition	243

Table 1: Number of verbatim per categories.

2.2 Dataset origin: Online posts

Yoomaneo is a free community platform open to all. It was created in 2020 by the company Ixiade.[5] Yoomaneo was created to build a database of individuals willing to participate in studies on Innovation. Ixiade is responsible for the recruitment of the participants of the studies, who are then invited to download the application. For our case, 755 responses or posts were extracted from Yoomaneo. These posts come from 4 different projects which focus on the evaluation of different innovative concepts in 3 different domains: health, well-being and electrical (2 projects).

Evaluation rules The collected posts were then given to 3 research fellows to evaluate. The evaluation procedure is similar to the one mentioned in section 2.1. Only the posts which received at least the same evaluation (same category when annotated) were kept. As a result, of the **755 evaluated**, **433** were assigned to the *motivation* class, **112** to the *barrier* class, **97** to the *condition* class, **65** were deemed unclassifiable, and **48** received no agreement. The distribution of the corpus is given in Table 2.

Classes	Number of verbatim
Barriers	112
Motivation	433
Condition	97

Table 2: Number of verbatim per categories.

3 Data Augmentation

Data amplification involves all the techniques for amplifying the amount of data available by adding slightly modified copies of the original data (Li et al., 2021) or artificially generating data from the original data through transformations (Taylor and Nitschke, 2018) with the goal of increasing the size of the dataset. It has been used in various fields such image classification (He et al., 2016), speech recognition (Park et al., 2019), etc. In this work, 4 different popular augmentation methods have been implemented and adapted for text classification for the French language (Bayer et al., 2021): synthetic noise (Feng et al., 2020; Belinkov and Bisk, 2017), synonym replacement (Wei and Zou, 2019; Feng et al., 2020; Coulombe, 2020), random trio techniques (Feng et al., 2020) and back-translation (Mercadier, 2020; Marivate and Sefara, 2020), (Feng et al., 2020), (Wei and Zou, 2019), and (Marivate and Sefara, 2020). To our knowledge, most of the mentioned techniques have only been applied to English data reviews and not on the type of data this work used: verbatim transcripts and online posts.

3.1 Synthetic noise

For each verbatim transcript in our training dataset, we randomly delete, insert and swap characters according to a replacement percentage rate. We produce for a verbatim transcript 5%, 10%, and 15% noise variations.

3.2 Random trio techniques

For random trio techniques, we randomly remove a word which is not a stopword, insert a random synonym of a word into a random position in the verbatim transcript and swap the position of two words with a percentage rate of 5% (5% of the words are changed).

3.3 Replacement methods

Lexical replacement approach is a technique that replaces a word or words in a text with similar words. Most works (Kolomiyets et al., 2011; Zhang et al., 2015) replace words in the original text with their synonyms using WordNet (Esuli and Sebastiani, 2007). Since we deal with French data, we used the lexical resource DBnary (Sérasset, 2012; Sérasset and Tchechmedjiev, 2014). DBnary is a large lexical resource which provides multilingual lexical data extracted from Wiktionary. The dataset contains extracts from 22 Wiktionary languages. We replace only adjectives, adverbs, verbs and nouns with a randomly chosen synonym of the same POS provided by DBnary. We use Stanza (Qi et al., 2020) for tagging.

[5] https://www.ixiade.com/

3.4 Back-translation

Back-translation (Sennrich et al., 2015) consists in translating a sentence from a source language to a target language. The sentence obtained after translation from the source language to the target language is then translated back into the source language. This approach makes it possible to obtain different variants of the same sentence. We use Deepl[6] translation service web to produce those new data for our training dataset. We used all the languages provided by Deepl, approximately 25 languages.

Method	Text
Original	Tout à fait. Après il peut y avoir une application pour les IPAD, et une autre pour les smart phone, c'est pas le même usage.
Synthetic Noise	Tout à faeit. Après il **put** y avoir **upne applictaiown** pour **lhes IaPAD**, et une autre pour les smart phone, **cv'est** pas le même usage.
Random trio	Tout à fait . Après il peut y avoir une **usage** pour les IPAD, et une autre pour les smart phone , c' est pas le même **application**.
Synonym replacement	Tout à fait . Après il peut y avoir une application pour les IPAD , et une autre pour les smart phone , c' est pas le même **emploi** .
Back-translation	C'est vrai. S'il existe une application pour iPad et une application pour smartphone, il ne s'agit pas du même travail.

Table 3: Example of a verbatim transcript and its variations using our augmentation methods. Changes are bolded.

4 Experimental Setup

We chose 4 data augmentation techniques and 2 Pretrained Models (FlauBERT and CamemBERT) for this experimental work.

4.1 Data splitting and augmentation

Methods	Training	Testing
Original	503	168
Synthetic Noise	1981	168
Random trio	8024	168
Synonym replacement	6822	168
Back-translation	11 236	168

Table 4: Overview of the augmented datasets for the verbatim dataset.

We divide our dataset into 3 subsets: train, dev and test (respectively 60%, 20%, 20%). We augment only the training set. Table 3 gives an example of verbatim transcripts generated using the different augmentation methods mentioned above. Table 4 and 5 gives an overview of the training size per augmentation method.

Methods	Training	Testing
Original	384	129
Synthetic Noise	1465	129
Random trio	5481	129
Synonym replacement	3854	129
Back-translation	7691	129

Table 5: Overview of the augmented datasets for the online posts dataset.

4.2 Pretrained Models and Finetuning

Model description. FlauBERT (Le et al., 2020) is a French BERT model. It was trained on 71 GB of French text corpus. The corpus consists of 24 subcorpora covering diverse topics and writing styles from formal and well-written text (e.g. Wikipedia and books).[7] CamemBERT is also a language model for French based on the RoBERTa (Liu et al., 2019) architecture pretrained on the French corpus OSCAR (Suárez et al., 2019) (138 GB) and CCNET (Wenzek et al., 2019) (135 GB). Both FlauBERT and CamemBERT were trained on the masked Language Modeling (MLM) task.

Architecture. For our task, we append the relevant predictive layer on top of CamemBERT's and FlauBERT's architecture. We fine-tune all the different models to follow the process described by Devlin et al. (2018) and followed by Le et al. (2020). The classification head for FlauBERT consists of the following layers, a dropout, a linear layer followed by the activation function tanh, a dropout and another linear layer. To obtain the probabilities for each class, the softmax function was used. The dimensions of the inputs of the linear layers are respectively equal to the size of the Transformer. For CamemBERT, the classification heads are the same as the ones described in Martin et al. (2019).

Parameters. As far as the hyperparameters are concerned, they are all fixed at the time of learning, with a batch size of 8 for all the architectures. The number of epochs is set to 5 and the learning rate to 5e-5 for the first epoch, then decreasing linearly. The AdamW (Kingma and Ba, 2014) optimizer is used.

[6] https://www.deepl.com/fr/translator

[7] http://www.gutenberg.org.

5 Results and Analysis

In this section, we present the results on our two test data. We compare the performance of FlauBERT with its competitor (CamemBERT). The metrics used to measure the performance of each method were the F1-score and the accuracy (F-micro). The F-score is used as metric since our data are imbalanced in order to observe the real performance of the model. The results are evaluated according to the amplification method used and the architecture used. Our baseline is the model without amplification.

5.1 FlauBERT

For FlauBERT, we use the 3 model sizes: FlauBERT BASE CASED (BC), FlauBERT BASE UNCASED (BU) and FlauBERT LARGE (L). Table 6 presents the size of data on which each model was trained.

Model	Parameters	Architecture	Training corpus
FlauBERT BASE CASED (BC)	138M	Base	24 corpora (71GB)
FlauBERT BASE UNCASED (BU)	137M	Base	24 corpora (71GB)
FlauBERT LARGE (L)	373M	Large	24 corpora (71GB)

Table 6: pre-trained model size for FlauBERT (Le et al., 2020).

TAD	Verbatim transcripts					
	FlauBERT Base Cased		FlauBERT Base Uncased		FlauBERT Large	
	accuracy	F1	accuracy	F1	accuracy	F1
0 - Baseline	0.482	0.217	0.500	0.267	0.589	0.538
1 - BT	0.667 +0.18	0.604 +0.39	0.690 +0.19	0.650 +0.38	**0.690** **+0.10**	**0.657** **+0.12**
2 - SR	0.589 +0.11	0.574 +0.36	0.649 +0.15	0.607 + 0.34	0.690 +0.10	0.641 +0.10
3 - RT	0.595 +0.11	0.558 +0.34	**0.714** **+0.21**	**0.683** **+0.42**	0.583 -0.01	0.411 -0.13
4 - NI	**0.673** **+0.19**	**0.591** **+0.37**	0.685 +0.18	0.641 +0.37	0.625 +0.04	0.514 -0.02

Table 7: FlauBERT: F1 and accuracy score for verbatim transcripts test data.

Table 7 presents the final accuracy and F1 on test set for the verbatim transcripts. The results show that FlauBERT BU performs better than the CASED model and the LARGE model, with an accuracy score of 0.714 and F1 score of 0.682. Overall, Back-translation and noise injection perform better for all the 3 models, with an average accuracy greater than 0.60. Huge improvement is observed with the F1 score for all the models, except for the case where FlauBERT LARGE is combined with random trio and Noise Injection. One reason may be that too much injection and replacement of words might have altered the semantic sense of the training data when augmenting it.

TAD	Online Posts					
	FlauBERT Base Cased		FlauBERT Base Uncased		FlauBERT Large	
	accuracy	F1	accuracy	F1	accuracy	F1
0 - Baseline	0,667	0,269	0,674	0,269	0,667	0,267
1 - BT	0,698 +0,03	0,514 +0,24	0,791 +0,12	0,660 +0,39	**0,822** **+0,15**	**0,750** **+0,48**
2 - SR	0,713 +0,05	0,582 +0,31	0,752 +0,08	0,621 +0,35	**0,829** **+0,16**	**0,733** **+0,47**
3 -RT	**0,736** **+0,07**	**0,648** **+0,38**	0,721 +0,05	0,515 +0,25	0,814 +0,15	0,723 +0,46
4 - NI	0,651 -0,02	0,484 +0,22	**0,798** +0,12	**0,714** +0,44	0,829 +0,16	0,729 +0,46

Table 8: FlauBERT: F1 and accuracy score for online posts test data.

Table 8 presents the results on the test set for online posts. The results show that FlauBERT L performs slightly better than the CASED model and the LARGE model, with an accuracy score greater than 0.80 for all the amplification methods. The best score is obtained with synonym replacement and FlauBERT L with an accuracy score of 0.829 and F1 of 0.733.

By comparing the results, we observe that the amplification methods combined with the different FLauBERT models improve the classification task for both test data. Nevertheless, the results are more significant on the online post data, with an accuracy above 0.80. This might be because they are somewhat similar to reviews or critics. Verbatim transcripts are quite particular since they come from oral dialogue which has been transcribed and revised. The classification models have somewhat more difficulties to classify those type of data compare to online posts, even though the accuracy is quite good (> 0.70 on verbatim transcripts test data). Random trio and synonym replacement are respectively the ones which produced the best score for the test data for verbatim transcripts and test data for online posts.

In the next section, we present the results obtained when using CamemBERT model.

5.2 CamemBERT

For CamemBERT, we used three model sizes which were introduced in Martin et al. (2019): CamemBERT BASE O for the model trained on the OSCAR corpus, CamemBERT BASE C for the model trained on the CCNET corpus and CamemBERT LARGE trained of the CCNET corpus.

Table 10 presents the final accuracy and F1 on test set for online posts. The results show that CamemBERT LARGE performs better than the BASE model, with an accuracy score of 0.756 and

Model	Parameter	Architecture	Training corpus
CamemBERT BASE O	110M	Base	corpus OSCAR (135 GB)
CamemBERT LARGE	335M	Large	corpus CCNet (135 GB)
CamemBERT BASE C	110M	Base	corpus CCNet (135 GB)

Table 9: Pre-trained models size for CamemBERT (Martin et al., 2020).

TAD	Verbatim transcripts					
	CamemBERT B (OSCAR)		CamemBERT B (CCNet)		CamemBERT L (CCNet)	
	accuracy	F1	accuracy	F1	accuracy	F1
0 - Baseline	0,482	0,217	0,607	0,458	0,494	0,318
1 - BT	0,696 +0,21	0,640 +0,42	**0,732** **+0,13**	**0,687** **+0,47**	0,673 +0,18	0,611 +0,39
2 - SR	**0,714** **+0,23**	**0,663** **+0,45**	0,702 +0,10	0,653 +0,44	0,649 +0,15	0,581 +0,36
3 - RT	0,714 +0,23	0,648 +0,43	0,655 +0,05	0,594 +0,38	**0,756** **+0,26**	**0,720** **+0,50**
4 - NI	0,685 +0,20	0,642 +0,43	0,667 +0,06	0,621 +0,16	0,714 +0,22	0,687 +0,37

Table 10: CamemBERT: F1 and accuracy score for Verbatim transcripts test data.

F1 score of 0.720. Random trio is the best performing method (acc.: 0.756) follow by the back-translation method (acc.: 0.732) and synonym replacement (acc.: 0.714).

TAD	Online Posts					
	CamemBERT B (OSCAR)		CamemBERT B (CCNet)		CamemBERT L (CCNet)	
	accuracy	F1	accuracy	F1	accuracy	F1
0 - Baseline	0,674	0,269	0,752	0,484	0,674	0,269
1 - BT	**0,783** **+0,11**	**0,68** **+0,41**	0,814 +0,06	0,755 +0,27	0,822 +0,15	0,771 +0,50
2 - SR	0,767 +0,09	0,672 +0,40	0,845 +0,09	0,759 +0,27	**0,868** **+0,19**	**0,801** **+0,53**
7 - RT	0,744 +0,07	0,666 +0,40	**0,853** **+0,10**	**0,780** **+0,30**	0,775 +0,10	0,708 +0,44
8 - NI	0,744 +0,07	0,580 +0,31	0,806 +0,05	0,731 +0,25	0,806 +0,13	0,624 +0,36

Table 11: CamemBERT: F1 and accuracy score for Online Posts test data.

Table 11 show that CamemBERT LARGE performs better than the BASE model, with an accuracy score of 0.868 and F1 score of 0.801. Random trio is the best performing method, follow by the back-translation method (acc.: 0.853) and synonym replacement (acc.: 0.783).

By comparing the results, we observe that augmentation methods used in this work clearly improved the performances for both CamemBERT and FlauBERT. Overall, CamemBERT performances are better than FlauBERT. Synonym replacement combined with CamemBERT LARGE is the best performing duo on verbatim and online posts test data. We also noted that performances on post online are better than on verbatim transcripts. One reason may be the type of data the pre-trained model were trained on. CCNET corpus were crawled from internet, so they may be more similar or linguistically closer to online posts than verbatim transcripts. A linguistic analysis of the data used to trained CamemBERT model may be interesting to conduct in order to explore the lingusitic similarities or differences with our datasets. In conclusion, the results are promising and clearly open up work prospects.

6 Conclusion

We have presented a work where we sought to compare the performances of two BERT models for French language on a three-class classification task . Firstly, we show that simple augmentation techniques used for text classification can be implemented and adapted for the datasets used in this work. Overall, we also obseved that CamemBERT model was better than FlauBERT for this task and the best amplification method was synonym replacement. For future works, we would like to use other pretrained language models for French such as XLNET, BERT multilingual, etc. In this paper, we just focus on comparing two French Variants. We also think exploring the linguistic features of our dataset in the training of the model may be interesting with the goal of evaluating their impact on the performance. Finally, we also think that trying to other amplification methods such as replacement via a language model may be interesting.

The data used in this work comes from a private enterprise, and we have not received their consent to share the dataset.

References

Markus Bayer, Marc-André Kaufhold, and Christian Reuter. 2021. A survey on data augmentation for text classification. *arXiv preprint arXiv:2107.03158*.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Claude Coulombe. 2020. *Techniques d'amplification des données textuelles pour l'apprentissage profond*. Ph.D. thesis, Télé-université.

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

Andrea Esuli and Fabrizio Sebastiani. 2007. Sentiwordnet: a high-coverage lexical resource for opinion mining. *Evaluation*, 17:1–26.

Steven Y Feng, Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. 2020. Genaug: Data

augmentation for finetuning text generators. *arXiv preprint arXiv:2010.01794*.

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.

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

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 2011. Model-portability experiments for textual temporal analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, volume 2, pages 271–276. ACL; East Stroudsburg, PA. En ligne à l'adresse : https://scholar.google.com/scholar?hl=fr&as_sdt=0%2C5&q=kolomiyets+synonym&btnG=.

Md Tahmid Rahman Laskar, Xiangji Huang, and Enamul Hoque. 2020. Contextualized embeddings based transformer encoder for sentence similarity modeling in answer selection task. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5505–5514.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. 2020. Flaubert: Unsupervised language model pre-training for french. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2479–2490, Marseille, France. European Language Resources Association.

Bohan Li, Yutai Hou, and Wanxiang Che. 2021. Data augmentation approaches in natural language processing: A survey. *arXiv preprint arXiv:2110.01852*.

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*.

Vukosi Marivate and Tshephisho Sefara. 2020. Improving short text classification through global augmentation methods. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 385–399. Springer.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. Camembert: a tasty french language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de La Clergerie, Djamé Seddah, and Benoît Sagot. 2019. Camembert: a tasty french language model. *arXiv preprint arXiv:1911.03894*.

Yves Mercadier. 2020. *Classification automatique de textes par réseaux de neurones profonds: application au domaine de la santé*. Ph.D. thesis, Université Montpellier.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

Yifan Peng, Shankai Yan, and Zhiyong Lu. 2019. Transfer learning in biomedical natural language processing: an evaluation of bert and elmo on ten benchmarking datasets. *arXiv preprint arXiv:1906.05474*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. corr abs/1802.05365 (2018). *arXiv preprint arXiv:1802.05365*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Gilles Sérasset. 2012. Dbnary: Wiktionary as a lmf based multilingual rdf network. In *Language Resources and Evaluation Conference, LREC 2012*.

Gilles Sérasset and Andon Tchechmedjiev. 2014. Dbnary: Wiktionary as linked data for 12 language editions with enhanced translation relations. In *3rd Workshop on Linked Data in Linguistics: Multilingual Knowledge Resources and Natural Language Processing*, pages 67–71.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Leibniz-Institut für Deutsche Sprache.

Luke Taylor and Geoff Nitschke. 2018. Improving deep learning with generic data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1542–1547. IEEE.

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.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2019. Ccnet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657. En ligne à l'adresse : https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf.

Semi-supervised Automated Clinical Coding Using International Classification of Diseases

Hlynur D. Hlynsson[1], **Steindór Ellertsson**[2], **Jón F. Daðason**[1], **Emil L. Sigurdsson**[2,3,4], **Hrafn Loftsson**[1]
[1] Department of Computer Science, Reykjavik University, Reykjavik, Iceland
[2] Primary Health Care Service of the Capital Area, Reykjavik, Iceland
[3] Department of Family Medicine, University of Iceland, Reykjavik, Iceland
[4] Development Centre for Primary Health Care in Iceland, Reykjavik, Iceland
{hlynurh, jond19, hrafn}@ru.is
steindor.ellertsson@gmail.com, emilsig@hi.is

Abstract

Clinical Text Notes (CTNs) contain physicians' reasoning process, written in an unstructured free text format, as they examine and interview patients. In recent years, several studies have been published that provide evidence for the utility of machine learning for predicting doctors' diagnoses from CTNs, a task known as ICD coding. Data annotation is time consuming, particularly when a degree of specialization is needed, as is the case for medical data. This paper presents a method of augmenting a sparsely annotated dataset of Icelandic CTNs with a machine-learned data imputation in a semi-supervised manner. We train a neural network on a small set of annotated CTNs and use it to extract clinical features from a set of un-annotated CTNs. These clinical features consist of answers to about a thousand potential questions that a physician might find the answers to during a consultation with a patient. The features are then used to train a classifier for the diagnosis of certain types of diseases. We report the results of an evaluation of this data augmentation method over three tiers of information that are available to a physician. Our data augmentation method shows a significant positive effect, which is diminished when an increasing number of clinical features, from the examination of the patient and diagnostics, are made available. Our method may be used for augmenting scarce datasets for systems that take decisions based on clinical features that do not include examinations or tests.

1 Introduction

When a patient consults a physician, communication is created in the patient's medical records. The physician notes down the patient's signs, symptoms, results of physical examination, the clinical thinking process, and if any diagnostic tests are warranted – in a free text format known as a Clinical Text Note (CTN). Then, the physician saves the diagnoses, using the International Classification of Diseases (ICD)[1] code, that they made during the consultation. Thus, each CTN contains free text, from which clinical features can be extracted, in addition to the ICD classification code.

Previous work has shown the benefits of training machine learning classifiers on clinical features for automated ICD coding (Liang et al., 2019; Ellertsson et al., 2021; Zhang et al., 2020; Pascual et al., 2021; Kaur et al., 2021; Blanco et al., 2021). Ellertsson et al. (2021) hand-annotated features in 800 CTNs and trained a classifier to predict ICD codes for one of four types of primary headache diagnoses. Liang et al. (2019) hand-annotated a significantly larger set, i.e. about 6,000 CTNs, for the purpose of training a classifier to predict various types of diseases, i.e. 55 ICD codes in total. Additionally, they developed a clinical feature extraction model (CFEM), for the purpose of automatically extracting features from the CTNs.

On its own, the CFEM is beneficial because it could solve the common clinical problem of getting a quick and comprehensive overview of a patient, when meeting a clinician for the first time. A clinician could search a patient's medical history with a question such as "Has the patient ever had a colonoscopy?". The ICD classifiers have, on the other hand, the potential of being integrated into a Clinical Decision Support System (CDSS), where they could, for example, predict if a physician should order an MRI for a patient when presented with a particular symptom, what kind of blood tests are warranted, or any other diagnostic test for that matter.

Generally, machine learning systems require large quantities of training data (Hlynsson et al., 2019) and ICD classifiers are no exception. In order to develop a high accuracy ICD classifier, without annotating large amount of CTNs, we experiment with a method of: 1) annotating a small subset of

[1] https://www.who.int/classifications/classification-of-diseases

the CTNs with question-answer pairs which are used for training the CFEM, and then 2) use the trained feature extractor to extract clinical features from samples out of a larger dataset of CTNs for training the classifier to predict one out of six ICD codes[2].

In prior work on ICD coding, classifiers have been trained on discharge summaries, after the patient has left the clinic (Liang et al., 2019; Zhang et al., 2020; Pascual et al., 2021; Kaur et al., 2021; Blanco et al., 2021). We instead focus on evaluating our model on stages in the primary health care pipeline where the recommendations of machine learning models would be the most effective. We thus introduce a novel three-tiered evaluation system that is designed to mirror the circumstances where ICD classification methods would actually be used and we evaluate our semi-supervised data augmentation method on these three tiers: 1) before the patient meets a physician, 2) after the physician performs the patient examination, and 3) after the physician has ordered diagnostic tests.

Our evaluation results show that the data augmentation method has a significant benefit for tier 1, i.e. before the patient meets a physician, but not for the other two.

2 Related Work

Liang et al. (2019) frame the problem of clinical feature extraction from CTNs as a question-answering task. Every clinical feature mentioned in a given CTN is marked, as well as the start and the end of the text span referring to a given clinical feature. A question is saved in the context of the text span, which contains the answer to that specific question. For example, given the text span "the patient has a fever", the question "Does the patient have a fever?" is saved with a binary value of 1. Out of 1.3 million CTNs from a single institution in China, Liang et al. (2019) annotated about 6,000 CTNs for training a CFEM, based on a Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) enriched with word embeddings. The feature extractor is trained on a batch of (CTN, question, text span) tuples as input with the goal of optimizing for the text span that contains the corresponding answer to the question in the given CTN. Thereby, the model learns to extract relevant clinical features from the questions put forward in the context of the CTN. Liang et al. (2019) used the CFEM to extract features from the whole set of un-annotated CTNs. The extracted features were then used to train a classifier, based on multi-class logistic regression, to predict an ICD code from a set of 55 codes.

Ellertsson et al. (2021) hand-annotated clinical features (in a similar manner as Liang et al.) in 800 CTNs from a common medical database of all primary care clinics in Iceland. Each CTN had an accompanying ICD code for one of four types of headache diagnoses. The resulting features (text spans) were then used to train a Random Forest classifier, for predicting one of the four possible ICD codes. Furthermore, they performed a retrospective study where the classifier was shown to outperform general practitioners on the four types of headache diagnostics.

In this paper, we expand upon the work of Ellertsson et al. The main difference between our work and theirs can be summarized as follows:

- We do not compare our ICD classifier to general practitioners.
- We hand-annotate questions-answers pairs in 2,422 CTNs, which includes a larger number of ICD codes, 42 in total (see Table 4 in the Appendix).
- Using the hand-annotated CTNs, we train CFEMs, based on Transformer models (Vaswani et al., 2017), for extracting clinical features, and compare them to a couple of LSTM models. These feature extractors are used to extract features from un-annotated CTNS as well as annotated CTNs.
- We perform a three-tiered evaluation of our classifiers on six of the ICD codes for pediatric (under 18) patients (see Table 5 in the Appendix).

Transformer-based models have rapidly become a popular choice for automated ICD coding. These models have been trained on CTNs in a fully end-to-end manner (Zhang et al., 2020; Pascual et al., 2021; Kaur et al., 2021; Blanco et al., 2021). A drawback of this approach is that physicians will often write down their hypothesized diagnoses which injects a serious bias to the data. We circumvent this problem by using one model for clinical feature extraction and another for clinical prediction.

[2]The ICD classes were chosen by doctors according to their perceived usefulness.

		Training Set	Validation Set	Test Set	Total
Adults	Total size	1700	199	220	2119
	Mean Age ± Std	45.33 ± 17.91	43.54 ± 17.86	44.24 ± 17.92	
	Min Age – Max Age	18.01 – 94.43	18.04 – 86.75	18.17 – 93.72	
Children	Total size	237	33	33	303
	Mean Age ± Std	10.01 ± 5.87	10.32 ± 5.82	9.39 ± 6.24	
	Min Age – Max Age	0.17 – 17.99	0.97 – 17.85	0.21 – 17.85	

Table 1: **Training data split statistics for the clinical feature extraction model.** The adult sets are 63% female and the child sets are 64% female. The different sizes of the adult validation and test sets came by to enforce a constraint of an equal proportion of notes corresponding to each ICD code within each set.

For example, a fully end-to-end machine learning model might learn to associate the qualitative comment by a physician "the patient probably has a migraine without aura" in a patient with a migraine-without-aura ICD code. Our method avoids this by creating a bottleneck of information, where only specific questions are being answered.

Our approach also opens the door for interpreting the results of the ICD classifier, as the importance of each input feature to the classifier can be visualized, for example by portraying input coefficients in the case of linear models (e.g. logistic regression) or plotting other interpretability metrics, such as SHAP values (Lundberg and Lee, 2017).

3 Approach

3.1 Data and annotation

We use the dataset from the same source as Ellertsson et al. (2021), i.e. from the Primary Health Care Service of the Capital Area (PHCCA) in Iceland. The dataset consists of 1.2 million CTNs, written in Icelandic, from 200 thousand unique patients that were collected in clinical consultations taking place from January 2006 to April 2020. Physicians are instructed not to write anything that can uniquely identify their patients in the notes, but we also used a combination of a parser for Icelandic (Þorsteinsson et al., 2019) as well as a regex command to remove any personally identifiable information, such as names, personal identification numbers and phone numbers. This dataset contains CTNs that have an associated ICD 10 code, but consist otherwise of unstructured text from which clinical features can be extracted.

In the same manner as described by Ellertsson et al., we reduced the full dataset by applying a filter which only keeps notes that contain any word from a medical keyword dictionary. From this reduced dataset, we randomly selected 2,422 notes which were manually annotated by a physician[3], resulting in question-answer pairs as described in Section 2.

As an example annotation, for a CTN containing the text "the patient is not coughing", one clinical feature is the pair consisting of the question "does the patient have a cough?" and the binary-valued answer "0", with the corresponding text span "not coughing". Some answers are continuous-valued, such as for the question "what is the patient's blood pressure?".

The number of clinical features that we use to train the extraction model to output is 942. This number represents the number of question-answer pairs in the dataset. There is typically a heavy class imbalance for each feature, where the binary questions have on average a 0.75 positive answer ratio, with a standard deviation of 0.2. The reason for this sweeping class imbalance is that physicians generally only ask questions that are relevant and with an affirmative answer.

For our three-tiered classifier evaluation, we define three strict subsets of these features, as described in Section 3.6. Each question is also paired with another binary variable which indicates whether an answer to that question can be found in the CTN or not.

The dataset is split into adults, that are 18 years old or older, and children. Within each age group, 80% of the dataset is allocated for training, 10% for development/validation, and hold out 10% for final testing (see Table 1). The split is stratified to ensure that each set has an equal proportion of sexes and ICD codes.

3.2 Pre-trained Transformer-based models

We compared four existing Transformer-based models in our experiments, based on the ELECTRA (Clark et al., 2020) and RoBERTa (Liu et al.,

[3]The annotator is a white Icelandic male physician in his thirties, specializing in general practice / family medicine.

2019) architectures. We evaluated an ELECTRA-small[4], ELECTRA-base[5] and two RoBERTa-base models[6,7] (consisting of 14M, 110M and 125M parameters, respectively). All models have been pre-trained on the Icelandic Gigaword Corpus (IGC) (Steingrímsson et al., 2018), which consists of approximately 1.69B tokens from genres such as news articles, parliamentary speeches, novels and blogs. For one of the RoBERTa models, which we refer to as RoBERTa+, the IGC was supplemented with texts obtained from online sources, increasing the size of the pre-training corpus to 2.7B tokens. The RoBERTa models were pre-trained for 225k steps with a batch size of 2k. Otherwise, all models were pre-trained using default settings (Daðason and Loftsson, 2022). The pre-training process and additional training data for the RoBERTa models is described in further detail by Snæbjarnarson et al. (2022).

3.3 LSTM architectures

For a baseline comparison, we created two LSTM models. The first one (LSTM 1) tokenizes and trains the embeddings from scratch, whereas the second one (LSTM 2) pre-processes the inputs with GloVe (Pennington et al., 2014) embeddings.

3.3.1 LSTM 1

The model splits up the tokenized input into question and content parts. The content, which contains text that may contain the answer, gets a 256-dimensional embedding and the question gets a 32-dimensional embedding. The reason for the difference in dimensionality is that there is a much greater variety in the composition of the contexts opposed to the standardized number of questions that is being processed. Each embedding is then passed to its own, uniquely parameterized two-layer bi-directional LSTM model, where each layer has 256 units.

The outputs from those two parts are then concatenated and used to 1) train a set of dense networks, where one is tasked with predicting whether an answer to the question can be found in the text and, if yes, the other dense network predicts the probability of the answer being affirmative (in the case of binary questions), and 2) predict the start and end indices of the tokens that mark the span of the answer in the context part.

3.3.2 LSTM 2

LSTM 2 has the same architecture as LSTM 1, except there is no embedding layer and the inputs have been processed by a pre-trained GloVe model. The GloVe embeddings[8] where pre-trained on the IGC.

3.4 Clinical feature extraction models

We fine-tuned the four Transformer-based models, mentioned in Section 3.2, on the hand-annotated data in order to develop a CFEM. The fine-tuning was carried out in the following manner: starting with the pre-trained transformers weights, the top layer was replaced with a randomly initialized network, and the whole system was then trained end-to-end for question-answering. We also trained the two LSTM models described in Section 3.3 from scratch for a CFEM comparison.

Each model learns to output the answer span for each question[9] as well as the probability of the answer being affirmative for binary-valued questions. The Transformer-based models were defined and trained using the Transformers (Wolf et al., 2019) and PyTorch libraries (Paszke et al., 2019) and the LSTM models were defined and trained using TensorFlow (Abadi et al., 2016).

3.5 Semi-supervised learning

Once our CFEMs were trained, we saved their outputs over all the CTNs (i.e. 2,422 annotated CTNs used for training and 750 randomly selected unannotated CTNs) to disk. The outputs define the matrix of independent variables X which is, along with the dependent variable array y of ICD codes, used to train our logistic regression ICD classifier (implemented in scikit-learn (Pedregosa et al., 2011)).

CTNs require expertise to interpret, which results in a high cost when labelling medical datasets. This is especially true for AI researchers that are

[4] https://huggingface.co/jonfd/electra-small-igc-is. CC-BY-4.0 license.

[5] https://huggingface.co/jonfd/electra-base-igc-is. CC-BY-4.0 license.

[6] https://huggingface.co/mideind/IceBERT. AGPL 3.0 license.

[7] https://huggingface.co/mideind/IceBERT-igc. AGPL 3.0 license.

[8] https://github.com/stofnun-arna-magnussonar/ordgreypingar_embeddings/tree/main/GloVe

[9] If the question is not answered in the CTN, the model outputs an impossible span in the text, which is technically implemented as starting at the 0^{th} token (a special "start" token) and ending on the 1^{st} token of the context.

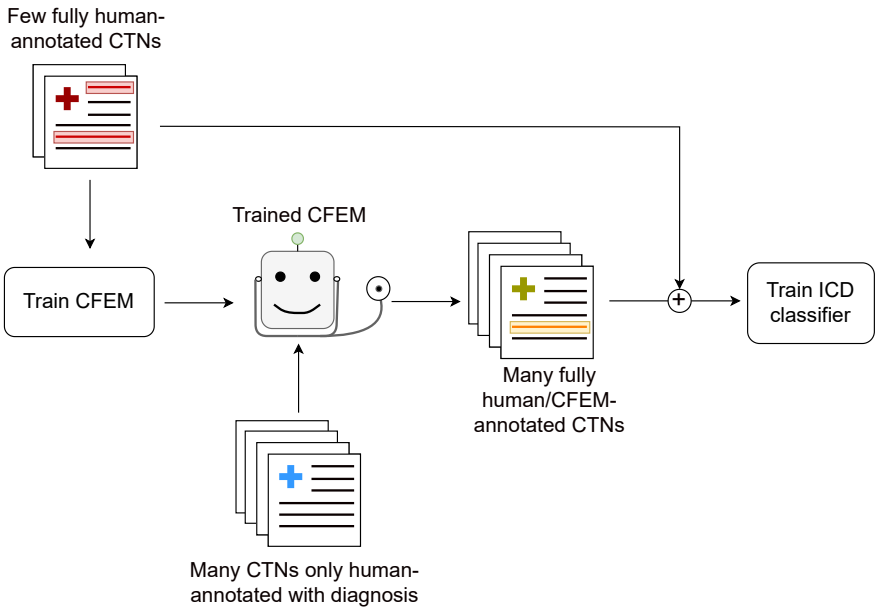


Figure 1: **Leveraging a Sparsely Annotated Dataset.** Our clinical feature extraction model learns to mark text spans (clinical features), containing an answer to a set of given clinical questions, from CTNs in which answer spans have been hand-annotated. The feature extractor is then used to extract answer spans – given the same set of questions – from a large set of CTNs that have diagnoses (ICD codes), but no marked answer spans. Finally, the extracted answer spans are used to train the ICD classifier. In this way, we make full use of a large set of CTNs that is only partly annotated and combine it with a much smaller set of human-annotated CTNs to learn automated ICD coding.

working with a language with much fewer resources than English (Blanco et al., 2021), such as Icelandic.

In our project, we have a large collection of CTNs, each of which is marked with a doctor's diagnosis, but does not contain answer spans for the set of questions for our clinical features. We input the un-annotated CTNs to a CFEM, that is trained on a much smaller subset of the data, to take advantage of the supervisory signal offered by the ICD code of each un-annotated CTN. This step keeps the interpretable clinical features and removes potential bias from the CTNs. This set of CTNs with imputed clinical feature values is then combined with our "gold standard" set of annotated CTNs, and both are used for training the ICD classifier (see Figure 1).

3.6 Three-tiered evaluation

To simulate the different stages of a physician's evaluation of a patient in real clinical circumstances, we limit the number of features that are available to the classifier at each stage:

- **Tier 1**: Before a patient meets with a physician. This includes the patient's main complaint, history, symptoms, and vital signs (420 features).
- **Tier 2:** After the patient has been examined by a physician (582 features).
- **Tier 3:** After results from diagnostics are available (608 features).

The full list of features is provided in the Appendix: Table 6 and Table 7 for tier 1, which are features that the patient could self-report. Tables 8 and 9 show the features for tiers 2 and 3, respectively. After tiers 2 and 3, decisions need to be taken regarding what further tests need to be ordered, for example imaging.

Note that our system could fit into a triage context at tier 1. The patient could fill out an online questionnaire and get recommendations depending on the results, for example, to go to the emergency room, to go the general physician, or maybe just rest at home with a set of self-care instructions.

4 Results and Discussion

4.1 Clinical feature extraction model training

The CFEMs were trained over three epochs on the subset of hand-annotated CTNs (see Table 1). For

the ELECTRA-base and RoBERTa-base transformers, each epoch took approximately eight hours on Cloud TPU v3 with eight cores, and half that for ELECTRA-small. The training took approximately three hours for each epoch for the LSTMs.

The RoBERTa+ model, which is pre-trained on the largest corpus, achieves the best results for all three metrics that we monitor (see Table 2): a span-based F_1-score, to evaluate the question-answering portion of the models, and the Matthews correlation coefficient (MCC) (Matthews, 1975; Chicco and Jurman, 2020) for the binary-valued clinical features (Binary MCC) and for predicting whether the question is answered in the text (Answered MCC).

We chose the MCC metric because it is appropriate for imbalanced data (Chicco, 2017) (see discussion of our data in Section 3.1) and it offers a suitable combination of the four confusion matrix metrics: true positives, true negatives, false positives and false negatives.

Note in Table 2 that the high F_1-scores are due to the fact that most questions were correctly predicted to be not answered in any given context. This could be due to the fact that the 15.8 GB corpus, which was used to train RoBERTa+, contains 33 MBs of medical texts. Although this is not a large proportion, it could be enough for the model to have learned transferable representations of medical vocabulary.

To our surprise, the ELECTRA-base model was outperformed by RoBERTa (both are trained on equal-sized corpora), even though ELECTRA has, previously, been shown to outperform RoBERTa on question-answering tasks (Clark et al., 2020).

The LSTM variation whose inputs were not pre-processed by a pre-trained GloVe model (LSTM 1) performed better according to the MCC metrics (but slightly worse according to the F_1-score) than the other (LSTM 2). We hypothesize that it is due to the fact that the pre-trained embeddings are not trained with any tokenization, but rather on whole words. The free-text style of doctor's notes can include words or abbreviations that are not defined for the GloVe embeddings.

4.2 ICD classifier training

4.2.1 Transformer vs. LSTM

After training and evaluating the CFEMs, we validated the data augmentation scheme described in Section 3.5. We used the best-performing models from each category, RoBERTa+ and LSTM 1, to extract the clinical features from the children's notes[10]. These features, along with their associated ICD codes, were then used to train the classifier.

	F_1	Bin. MCC	Answer MCC
RoBERTa+	**0.993**	**0.846**	**0.872**
RoBERTa	0.991	0.780	0.823
ELECTRA-base	0.987	0.656	0.729
ELECTRA-small	0.982	0.553	0.650
LSTM 1	0.975	0.331	0.327
LSTM 2	0.979	0.313	0.257

Table 2: **Feature extraction model evaluation results.** Question-answering metrics and evaluation results for each clinical feature extraction model on the test set. *Binary MCC* measures the classification accuracy of the binary-valued features and *Answer MCC* measures the accuracy of predicting whether a feature is answerable in the text.

Table 3 shows the diagnostic metrics of the classifier for tier 3 depending on the feature extractor. Using RoBERTa+ yielded a higher weighted average for all diagnostic metrics compared to LSTM 1.

4.2.2 Qualitative analysis

To verify that the relationship between our features and the outputs of our models matches our clinical intuition, we use SHAP (Shapley additive explanation) values (Shapley, 1953) to show the impact of each feature in the prediction of our logistic regression classifier, trained on the features in tier 3 extracted by RoBERTa+.

The feature importance plot is shown in Figure 2. We see, for example, that the top four features are headache-related features and contribute to classifying a CTN as Tension-type headache, migraine with- and without aura. The two top features after that involve the doctor doing a physical examination of the patient's lung and contribute to predicting whether the patient has pneumonia or bronchitis. The sixth most impactful feature is then the result of an examination of the patient's ear, the result of which contributes to the diagnosis of Otitis media (a disease of the middle ear).

4.2.3 Data augmentation experiment

In the next set of experiments, we investigated the effect of augmenting a data set consisting of 303 human-labeled childrens's CTNs with a varying

[10]Due to time constraints, our evaluation of the data augmentation method is limited to only using the children CTNs.

Condition	*RoBERTa+* F_1-score	MCC	TPR	TNR	*LSTM 1* F_1-score	MCC	TPR	TNR
Migraine without aura	0.40	0.36	0.33	0.97	0.00	0.00	0.00	1.00
Migraine with aura	0.67	0.70	0.50	1.00	0.40	0.36	0.33	0.97
Tension-type headache	0.94	0.89	1.00	0.88	0.86	0.73	1.00	0.71
Otitis media, unspecified	0.00	0.00	0.00	1.00	0.57	0.60	1.00	0.90
Bacterial pneumonia	0.86	0.83	1.00	0.93	0.75	0.75	0.60	1.00
Acute bronchitis	1.00	1.00	1.00	1.00	0.33	0.29	0.25	0.97
Weighted average	**0.81**	**0.78**	**0.85**	**0.85**	0.64	0.56	0.70	0.70

Table 3: **Detailed ICD classification metrics.** Per-class metrics for clinical diagnosis prediction when a logistic regression classifier is trained on features extracted from CTNs by either our RoBERTa+ transformer or the baseline LSTM 1 model. MCC is the Matthews correlation coefficient, TPR is the true positive rate and TNR is the true negative rate.

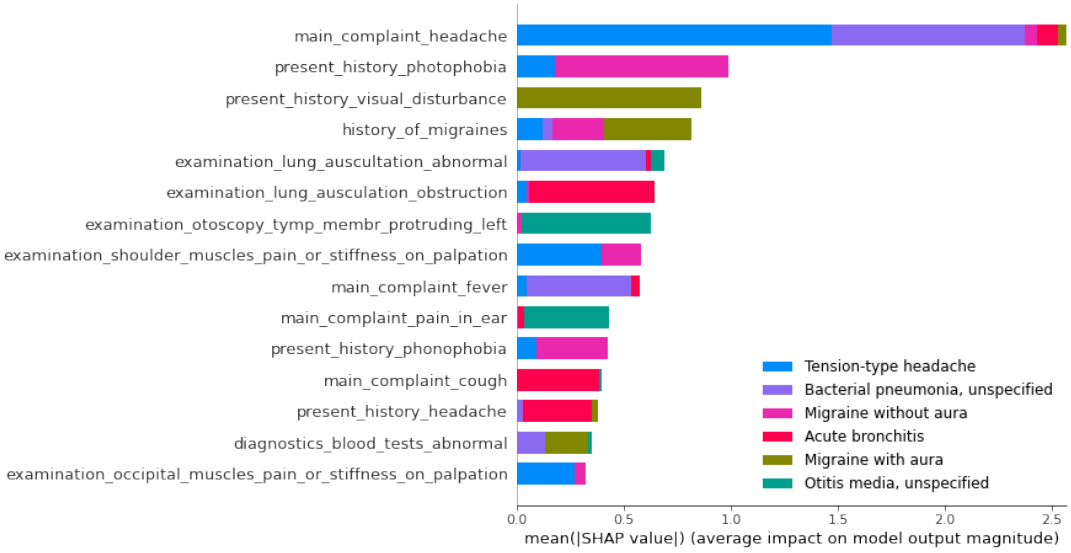


Figure 2: **Feature importance plot.** The features are scored by their SHAP values. The size of the colored bar in each feature's row indicates the contribution of that feature to predicting the disease with the corresponding color.

number of machine-labeled children's CTNs for the purpose of training an ICD classifier.

We trained logistic regression classifiers using 5-fold cross-validation over the whole children set. Each classifier had L1 regularization with the inverse regularization parameter of $C = 0.2$, which was found to give good classification performance in early tests. We chose not to do hyper-parameter tuning as the scope of this project is not to get the best possible classifier in this context, but rather investigate the data augmentation and the three-tiered evaluation. The results are shown in Figure 3.

There is a clear benefit for using the data augmentation method in tier 1, but it looks rather harmful for tiers 2 and 3. We hypothesize that this is due to the fact that the classifiers place a high importance on the outcome of examination (tier 2) and test (tier 3) related features, making the classifiers more sensitive to prediction errors for these feature.

5 Conclusions and Future Work

Our results show that training a CFEM on a small annotated subset of CTNs and use it to extract features from samples out of a larger, un-annotated dataset can increase the performance of an ICD classifier. However, the effect is only positive and significant in the context before a patient has been examined by the physician.

A future line of work is to further validate different classifiers by performing prospective studies which allow us to get insight into how the classifier performs in real clinical situations. This can be done by integrating the classifier into a CDSS, where a patient can log into a secure portal, at home or at a medical institution, and answer targeted questions regarding their symptoms. The

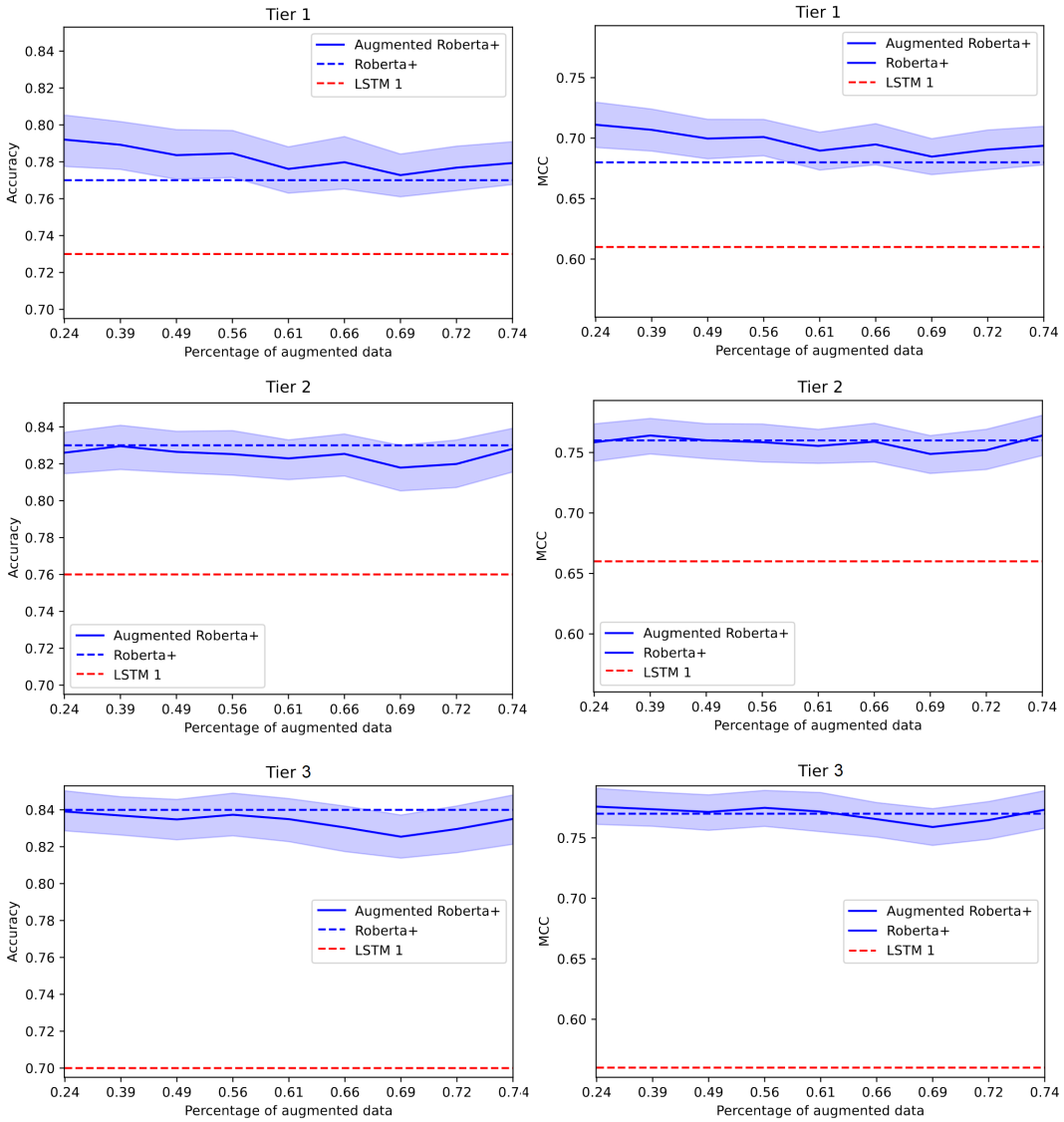


Figure 3: **Data Augmentation Results.** Each classifier is trained on fixed set of hand-annotated clinical features, in addition to a varying number of features automatically extracted by the RoBERTa+ model, i.e. machine-labeled features. There are 237 hand-annotated CTNs in each training set and each step along the x-axis adds 75 machine-labeled CTNs. Each point in the augmented curves shows the cross-validated metrics (accuracy in the left column and MCC in the right column) averaged over 20 random subsets of machine-labeled points that are added to the training set and the error band (the colored area around the Augmented Roberta+) signifies the 95% confidence intervals. The dashed lines indicate the performance of the classifiers trained only on hand-annotated data.

CDSS could build a list of differential diagnoses, recommend further diagnostics based on the patients symptoms, and then write out the CTN for the clinician. This does not disturb the clinical workflow, saves time for medical staff and potentially allows a much more detailed history taking, compared to the often time constrained clinician. This is important in all outpatient care, both public and private, since this kind of system has the potential to save money, increase the effectiveness and revenue for private clinics without losing the quality of care.

Acknowledgements

This work was funded by the Icelandic Strategic Research and Development Programme for Language Technology 2021, grant no. 200106-5301, and with Cloud TPUs from Google's TPU Research Cloud (TRC).

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283.

Alberto Blanco, Sonja Remmer, Alicia Pérez, Hercules Dalianis, and Arantza Casillas. 2021. On the Contribution of Per-ICD Attention Mechanisms to Classify Health Records in Languages with Fewer Resources than English. In *RANLP 2021: Recent Advances in Natural Language Processing, 1-3 Sept 2021, Varna, Bulgaria*, pages 165–172. Association for Computational Linguistics.

Davide Chicco. 2017. Ten quick tips for machine learning in computational biology. *BioData mining*, 10(1):1–17.

Davide Chicco and Giuseppe Jurman. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *arXiv preprint arXiv:2003.10555*.

Jón F. Daðason and Hrafn Loftsson. 2022. Pre-training and Evaluating Transformer-based Language Models for Icelandic. In *Proceedings of the 13th International Conference on Language Resources and Evaluation (LREC 2022)*, Marseille, France.

Steindor Ellertsson, Hrafn Loftsson, and Emil L. Sigurdsson. 2021. Artificial intelligence in the GPs office: a retrospective study on diagnostic accuracy. *Scandinavian Journal of Primary Health Care*, 39(4):448–458.

Hlynur Hlynsson, Alberto Escalante-B., and Laurenz Wiskott. 2019. Measuring the Data Efficiency of Deep Learning Methods. In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*. SCITEPRESS - Science and Technology Publications.

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

Rajvir Kaur, Jeewani Anupama Ginige, and Oliver Obst. 2021. A Systematic Literature Review of Automated ICD Coding and Classification Systems using Discharge Summaries. *arXiv preprint arXiv:2107.10652*.

Huiying Liang, Brian Y Tsui, Hao Ni, Carolina CS Valentim, Sally L Baxter, Guangjian Liu, Wenjia Cai, Daniel S Kermany, Xin Sun, Jiancong Chen, et al. 2019. Evaluation and accurate diagnoses of pediatric diseases using artificial intelligence. *Nature medicine*, 25(3):433–438.

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*.

Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*, 30.

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Vilhjálmur Þorsteinsson, Hulda Óladóttir, and Hrafn Loftsson. 2019. A Wide-Coverage Context-Free Grammar for Icelandic and an Accompanying Parsing System. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1397–1404.

Damian Pascual, Sandro Luck, and Roger Wattenhofer. 2021. Towards BERT-based Automatic ICD Coding: Limitations and Opportunities. *arXiv preprint arXiv:2104.06709*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32:8026–8037.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Lloyd S Shapley. 1953. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton.

Vésteinn Snæbjarnarson, Haukur Barri Símonarson, Pétur Orri Ragnarsson, Svanhvít Lilja Ingólfsdóttir, Haukur Páll Jónsson, Vilhjálmur Þorsteinsson, and Hafsteinn Einarsson. 2022. A Warm Start and a Clean Crawled Corpus - A Recipe for Good Language Models. In *Proceedings of the 13th International Conference on Language Resources and Evaluation (LREC 2022)*, Marseille, France.

Steinþór Steingrímsson, Sigrún Helgadóttir, Eiríkur Rögnvaldsson, Starkaður Barkarson, and Jón Guðnason. 2018. Risamálheild: A Very Large Icelandic Text Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

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, et al. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv preprint arXiv:1910.03771*.

Zachariah Zhang, Jingshu Liu, and Narges Razavian. 2020. BERT-XML: Large Scale Automated ICD Coding Using BERT Pretraining. *arXiv preprint arXiv:2006.03685*.

A Appendix

ICD code	Description
G43.0	Migraine without aura
G43.1	Migraine with aura
G44.0	Cluster headaches and other trigeminal autonomic cephalgias
G44.2	Tension-type headache
G44.4	Drug-induced headache, not elsewhere classified
G45.9	Transient cerebral ischemic attack, unspecified
H66.0	Acute suppurative otitis media
H66.9	Otitis media, unspecified
I10	Essential (Primary) Hypertension
I63.0+	Cerebral infarction
I63.1	Cerebral infarction
I63.2+	Cerebral infarction due to unsp. occl. or stenosis of precerebral arts.
I63.3	Cerebral infarction due to thrombosis of cerebral arts.
I63.4	Cerebral infarction due to embolism of cerebral arteries.
I63.5	Cerebral infarction due to unsp. occl. or stenosis of cerebral arts.
I63.6	Cerebral infarction due to cerebral venous thrombosis, nonpyogenic
I63.8	Other cerebral infarction
I63.9	Cerebral infarction, unspecified
I84	Haemorrhoids
J00	Acute nasopharyngitis [common cold]
J01	Acute sinusitis
J01.0	Acute maxillary sinusitis
J01.9	Acute sinusitis
J02.0	Streptococcal pharyngitis
J03.0	Streptococcal tonsillitis
J03.9	Acute tonsillitis
J05.0	Acute obstructive laryngitis
J10.1	Influenza due to other identified influenza virus w/ other resp. manifs.
J11.1	Influenza with other resp. manifs., virus not identified
J12.9	Viral pneumonia, unspecified
J15	Bacterial pneumonia, not elsewhere classified
J15.7	Pneumonia due to Mycoplasma pneumoniae
J15.8	Pneumonia due to other specified bacteria
J15.9	Bacterial pneumonia, unspecified
J20.9	Acute bronchitis
J44.1	Chronic obstructive pulmonary disease with (acute) exacerbation
J44.9	Chronic obstructive pulmonary disease, unspecified
J45.0	Predominantly allergic asthma
J45.9	Asthma, unspecified
M54.1+	Radiculopathy
M54.5+	Low back pain
S83.2	Tear of meniscus, current injury

Table 4: ICD codes associated with notes used during training of the clinical feature extraction model.

ICD code	Description
G43.0	Migraine without aura
G43.1	Migraine with aura
G44.2	Tension-type headache
H66.9	Otitis media, unspecified
J15.9	Bacterial pneumonia, unspecified
J20.9	Acute bronchitis

Table 5: ICD codes associated with notes using during classifier training.

History of migraines	History smoking	History smoking package years v	History of wiplash	History of alcoholism	History is regularly active	History of bells palsy
History of stroke	History of hypertension	History of migraines with descr	History active use alcohol mode	History active substance abuse	History of accident motor vehic	History of cigarette smoking
History of head trauma	History smoking active time	History smoking packages per da	History known allergy	History of cluster headache	History of depression	History of anxiety
History allergy pollen	History of fibromyalgia	History of headaches	History allergy penicillin	History of osteoarthritis	History of epilepsy	History of copd
History of pulmonary cancer	History of ischemic heart disea	History of diabetes mellitus 2	History of polio	History of hyperlipidemia	History of lupus	History of asthma
History of sinusitis	History of diabetes mellitus	History of hypothyrosis	History of palpitations	History of adhd	History of lower back disc prot	History of atrial fib flutter
History known medical allergy	History of chrons ds	History of dvt	History of bipolar disease	History allergy sulfa	History tonsillectomy	History appendectomy
History hepatitis c	History of prescription drug ab	History of streptococcal pharyn	History of sleep apnea	History of pad	History of lobectomy	History of heart failure
History of gastritis	History unilat or bilat catarac	History of pulmonary embolus	History of c section	History of reflux	History of ca mammae	History allergy tramadol
History o2 at home	History of heart attack	History of cabg surgery	History of renal cancer	History artificial heart valve	History has pacemaker	History copd gold stage
History cardiac catherization d	History of nephrectomy	History of cholecystectomy	History active substance abuse	History of psoriasis	History cancer prostata	History sick sinus
History of gerd	History hiatal hernia	History of recurrent otitis med	History of being prematurely bo	History has one kidney	History of diabetes mellitus 1	History hysterectomi
History of benign prostate hype	History of recurrent pneumonia	History of reumoatoid arthritis	History allergy morphine	History of pulmonary hypertensi	History joint proteses	History smoking time since quit
History of kidney stones	History of diverticulitis	History of gout	History allergy ibuprofen	History of substance abuse	History multiple sclerosis	History inactive subtance abuse
History active cancer	History of recurrent cystitis	History of chronic renal failur	History of aortic stenosis	History of chest pain	History breast wedge excision	History of glaucoma
History of colitis ulcerosa	History of diverticulosis	History of chronic diarrhea	History of compression fracture	History of spinal stenosis	History of dementia	History of heart valve disease
History is blind or close to bl	History of parkinsons disease	History has a single lung	History smoking stop year	History of tia	History of eczema	History of iron deficiency
History of backpain	History allergy voltaren	History allergy brown band aid	History of pneumonia	History of osteoporosis	History of accident type unkown	Present history nausea
Present history tinnitus	Present history shoulder and ba	Present history vomiting	Present history visual disturba	Present history aura	Present history photophobia	Present history recent head tra
Present history runny nose	Present history bulbar conjunct	Present history phonophobia	Present history chest pain	Present history dyspnea	Present history fever	Present history limb numbness
Present history dizziness	Present history recedes to quie	Present history facial or head	Present history limb reduced fo	Present history head trauma	Present history ptosis	Present history malaise
Present history diplopia	Present history flashing lights	Present history using analgesic	Present history aphasia	Present history nasal congestio	Present history wakes up with s	Present history is hearing chan
Present history abdominal pain	Present history feeling unbalan	Present history vertigo	Present history dizzyness on he	Present history syncope	Present history dysphasia	Present history memory problem
Present history visual disturba	Present history visual disturba	Present history headache	Present history insomnia	Present history diarrhea	Present history is pregnant	Present history pregnancy durat
Present history ear muffled bil	Present history back pain	Present history blood in stool	Present history common cold sym	Present history sore throat	Present history dysphagia	Present history cough
Present history melena	Present history dysuria	Present history snores	Present history nose bleeding	Present history palpitations	Present history treated by phys	Present history flulike symptom
Present history mate has notice	Present history body bone muscl	Present history pain in ear	Present history has iron defici	Present history has physiothera	Present history obstipation	Present history pain appears or
Present history sputum excretio	Present history chest tightness	Present history chest tightness	Present history two kinds of he	Present history chills	Present history pain after fall	Present history pain in chest o
Present history sputum excretio	Present history pleural pain	Present history recent fever	Present history recently finish	Present history ear muffled	Present history fluid out of ea	Present history tympanostomy tu
Present history involuntary los	Present history arrives with am	Present history has not taken t	Present history reduced fluid i	Present history reduced food in	Present history worst headache	Present history pain in joints

Table 6: **Tier 1 features.** Part 1 of 2.

Present history hemoptysis	Present history pollakisuria	Present history recent surgery	Present history reduced urine o	Present history uneasy	Present history itching	Present history pain in shoulde
Present history recent long fli	Present history night sweats	Present history pain in calve a	Present history recent stimulan	Present history is trying to qu	Present history dizziness nauti	Present history bed ridden bc o
Present history referred from p	Present history macroscopic hem	Present history inflammation in	Present history throat burn	Present history is using immuno	Present history abdominal pain	Present history vitals taken af
Present history urine incontine	Present history recently diagno	Present history repeated airway	Present history recently diagno	Present history increased o2 ne	Present history bedridden	Present history recently diagno
Present history increased leg e	Present history burn in throat	Present history chest pain resp	Present history trouble breathi	Present history feels feverish	Present history urinary stenosi	Present history hard to breath
Present history nocturnal dyspn	Present history unable to use r	Present history confusion	Present history hoarseness	Present history increased sweat	Present history visual field ab	Present history increased clums
Present history symptoms have r	Present history lower extremiti	Present history trauma	Present history unlike self acc	Present history hemi symptoms	Present history cough at night	Present history pain caused by
Present history back pain thora	Present history back pain lumbo	Present history neck pain	Present history pain in single	Present history pain in groin	Present history pain reduction	Present history saddle numbness
Present history morning stiffne	Present history leg length disc	Present history fecal incontine	Present history pain reduction	Present history unable to work	Present history pain in buttock	Present history pain increases
Family history migraine	Family history hypertension	Family history heart disease	Family history hemochromatosis	Family history multiple scleros	Family history stroke	Family history of brain tumour
Family history of diabetes mell	Family history of deep venous t	Family history of brain aneurys	Family history of lower back di	Pain character pulsating	Pain pain killers work well	Pain onset
Pain vas value	Pain stability	Pain character heavy	Pain character tension	Pain character pressure	Pain character sting	Pain radiation neck
Pain disturbs sleep	Pain radiation teeth	Pain over maxillary sinuses	Pain vas variable	Pain worsens or gets better wit	Pain radiation to left arm	Pain radiation to jaw
Pain radiation to right arm	Pain vas worst value	Pain radiation to back	Pain appears or worsens on vals	Pain changes with food intake	Pain appears or worsens when co	Pain over frontal sinuses
Pain appears or worsens with po	Pain appears or worsens on layi	Pain location thorax back	Pain character electrical	Pain appears or worsens when st	Pain appears or worsens when si	Symptom start a few weeks ago
Symptom duration 24 hrs or more	Symptom frequency a few times p	Symptom start a few days	Symptom duration one hour or le	Symptom frequency a few times p	Symptom start a few months	Symptom trigger
Symptom localisation on the rig	Symptom duration a few hours	Symptom start a year or longer	Symptom localisation on the lef	Symptom frequency every day	Symptom frequency a few times p	Symptom frequency a few times a
Symptom start a few hours	Symptom duration is variable	Symptom frequency is variable	Symptom localisation goes betwe	Symptom duration a few minutes	Symptom nsaids work well	Symptom duration a few seconds
Symptom start a specific date	Main complaint headache	Main complaint prescription ren	Main complaint nose bleeding	Main complaint visual disturban	Main complaint allergy	Main complaint dizziness
Main complaint multiple problem	Main complaint syncope	Main complaint numbness in head	Main complaint back pain	Main complaint pain in knee	Main complaint nausea	Main complaint common cold symp
Main complaint aphasia	Main complaint high blood press	Main complaint malaise	Main complaint pain around sing	Main complaint vomiting	Main complaint outside lesion o	Main complaint abdominal pain
Main complaint chest pain	Main complaint numbness in limb	Main complaint dyspnea	Main complaint physiotherapy re	Main complaint depression and o	Main complaint car accident	Main complaint shoulder and bac
Main complaint shoulder problem	Main complaint pain in joints	Main complaint certificate	Main complaint referral to spec	Main complaint constipation	Main complaint needs prescripti	Main complaint is pregnant
Main complaint cough	Main complaint re assessment	Main complaint resp. symp	Main complaint fever	Main complaint pharyngitis	Main complaint pain in chest or	Main complaint pain in ear
Main complaint maxillary skin i	Main complaint body bone muscle	Main complaint external tumour	Main complaint trouble breathin	Main complaint sputum excretion	Main complaint sputum excretion	Main complaint chest tightness
Main complaint pleural pain	Main complaint chills	Main complaint impared consciou	Main complaint dysuria	Main complaint migraine	Main complaint palpitations or	Main complaint asthma exacerbat
Main complaint nasal congestion	Main complaint limb reduced for	Main complaint face reduced for	Main complaint feeling unbalanc	Main complaint slurry speech	Main complaint pain in hip	Main complaint pain in lower ex
Main complaint pain in buttock	Cough disturbs sleep	Cough accompanying abdominal pa	Cough barking	Heart rate value	Heart rate value self measureme	Heart rate left side value
Heart rate value self measureme	Respiratory frequency value	Oxygen saturation value	Temperature value	Temperature at home value	Blood pressure value	Blood pressure value self measu

Table 7: **Tier 1 features.** Part 2 of 2.

Examination lung auscultation a	Examination proprioception abno	Examination is obese	Examination palpable neck lymph	Examination heart auscultation	Examination systolic heart murm	Examination systolic heart murm
Examination abnormal or absent	Examination abnormal neurologic	Examination abnormal neurologic	Examination pronator drift	Examination positive babinsky	Examination finger nose test ab	Examination rhomberg abnormal
Examination abnormal heel to to	Examination abnormal gait	Examination abnormal or asymmet	Examination abnormal neurologic	Examination abnormal sensation	Examination neurological reflex	Examination is blood pressure e
Examination abnormal abdominal	Examination pupils abnormal	Examination neck stiffness	Examination generally sick look	Examination spontant nystagmus	Examination disturbed eye movem	Examination dix hallpike positi
Examination pain with sinus pal	Examination occipital muscles p	Examination slurry speech	Examination is line walking abn	Examination vitals are abnormal	Examination edema	Examination audible carotis bru
Examination abnormal or reduced	Examination abnormal force uppe	Examination abnormal force lowe	Examination shoulder muscles pa	Examination restricted neck mov	Examination is overweight	Examination nystagmus
Examination abnormal or asymmet	Examination lung auscultation c	Examination abnormal sensation	Examination abnormal sensation	Examination abdomen epigastrium	Examination abdomen llq pain on	Examination abdomen rlq pain on
Examination lung auscultation w	Examination lung auscultation r	Examination mouth throat abnorm	Examination reflexes patella ab	Examination abnormal or asymmet	Examination ataxia	Examination spurlings test posi
Examination lasegue positive si	Examination heart rate irregula	Examination grasset test abnorm	Examination lymph nodes palpabl	Examination otoscopy abnormal b	Examination dysdiadochokinesia	Examination ram normal
Examination pain on scm palpati	Examination fundoscopy abnormal	Examination visual field abnorm	Examination renal pain on percu	Examination abdomen ruq pain on	Examination abdomen pain on rel	Examination otoscopy cerumen bi
Examination weak to see	Examination reflexes achilles a	Examination reflexes triceps ab	Examination reflexes biceps abn	Examination thyroid palpation a	Examination costal intercostal	Examination tonsils enlarged
Examination tonsils pus	Examination lumbosacral pain on	Examination face reduced force	Examination language understand	Examination reflexes brachiorad	Examination clonus	Examination rash on body
Examination pain on palpation b	Examination otoscopy redness in	Examination pain or no pulse on	Examination pain on palpation p	Examination capillary refill ti	Examination visible petechiae	Examination lung auscultation c
Examination lung auscultation c	Examination distal vascular sta	Examination lung auscultation p	Examination otoscopy visible ef	Examination tymp tube not in pl	Examination tymp tube not in pl	Examination otoscopy pus in ear
Examination lung auscultation r	Examination trismus	Examination lung ausculation ob	Examination neck venous stasis	Examination abdomen luq pain on	Examination struggles with brea	Examination otoscopy tympanic m
Examination abdomen suprapubic	Examination lung auscultation c	Examination lung auscultation c	Examination abdomen murphys sig	Examination systolic heart murm	Examination o2 value	Examination venous stasis derma
Examination skin pallor	Examination tonsils cryptic	Examination otoscopy visible va	Examination otoscopy tymp membr	Examination lung auscultation c	Examination calves pain on palp	Examination tympanic membrane r
Examination stridor	Examination using abdominal mus	Examination otoscopy tympanic m	Examination otoscopy visible ef	Examination fontanella abnormal	Examination otoscopy tymp membr	Examination otoscopy tymp membr
Examination tympanic membrane r	Examination otoscopy tympanic m	Examination central cyanosis	Examination abdomen diffuse pai	Examination lung auscultation m	Examination calves redness or i	Examination pitting edema lower
Examination tympanic membrane r	Examination nose alae flutter	Examination lung deafness on pe	Examination lymph nodes palpabl	Examination pus in eyes bilat	Examination cold extremeties by	Examination mucous membranes dr
Examination abdomen visible her	Examination intestinal sounds a	Examination neglect present	Examination sbs value	Examination soft gum abnormal a	Examination lumbo sacral pain o	Examination signs of scoliosis
Examination hip reduced range o	Examination pain on palpation t	Examination restricted movement	Examination trouble walking on	Examination signs of kyphosis	Examination signs of abnormal l	Examination pain on palpation g

Table 8: **Tier 2 features.** This tier also includes the previous tier's features.

Blood tests tnt value	Blood creatinine value	Blood alat value	Blood total cholesterol value	Blood hdl value	Blood pressure left upper arm v	Blood mcv value
Blood tsh value	Blood wbc value	Blood neutrophils value	Blood tests tnt 2 value	Blood d dimer value	Blood bnp value	Blood astrup abormal
Blood inr value	Diagnostics blood tests abnorma	Diagnostics blood tests tnt ele	Diagnostics blood status abnorm	Diagnostics blood tests d dimer	Diagnostics blood glucose value	Diagnostics blood esr value

Table 9: **Tier 3 features.** This tier also includes the two previous tiers' features.

Recent Advances in Long Documents Classification Using Deep-Learning

Muhammad Al-Qurishi
Elm Company,
Research Department,
Riyadh 12382, Saudi Arabia,
mualqurishi@elm.sa

Abstract

Long document classification is one of the most challenging linguistic processing tasks. Recently, Recent deep-learning models such as the transformers have proven to perform this task with a high-level of success. Such models typically include the self-attention mechanism, which makes the calculations extremely complex as document length increases. In order to unlock the use of the most accurate document classification tools on a wider range of document types and make the use of methods based on self-attention practically feasible, it's necessary to introduce some innovations that facilitate better scaling. In this work we provide a quick and concise survey of recent research work in the area of long documents classification using deep-learning techniques. The advantages and disadvantages of these methods have been discussed along with some directions that may be useful in future research.

1 Introduction

Modern deep learning models for semantic analysis can achieve impressive results after they are trained on very large datasets, gaining the ability to generate highly accurate predictions about content they haven't seen before. However, their capacity to capture the relationships between words and sentences is dependent on statistical operations that grow more complex as the length of the text sequence is increased (Tay et al., 2020). Effectively, this renders many of the best methods nearly useless from a practical standpoint and necessitates development of tools that can realistically process long documents and return reliable results within a reasonable timeframe. In particular, methods that rely on the attention mechanism (BERT (Devlin et al., 2018) and its derivatives) tend to become computationally demanding when working with long text documents (Vaswani et al., 2017).

Resolving this problem would allow for much wider use of document classification algorithms and would potentially allow large organizations in many different sectors to manage their administrative burden more efficiently. This is why researchers are looking into different possibilities for updating the existing algorithms, implementing changes aimed specifically at improving performance with long text, and testing hybrid approaches that offer better ratio between training/inference time and accuracy of prediction (Wagh et al., 2021; Tay et al., 2020). Their efforts are going in different directions and at present time it's unclear which approach might offer the best chances to overcome the current limitations, but there are already some promising findings that could hint towards sustainable solutions.

In this research paper we are trying to introduce a quick and concise summaries of the recent research papers published in the area of deep learning that trying to solve the problem of long document classification. Then we compare them from several perspectives such as: used method, authors objectives, performance and datasets. Finally, we conclude this review with a discussion of some ideas and suggestion that might become the basis for a new research in this area.

2 Long Document Classification Techniques

In this section we categorize the techniques and methods that have been used for long document classification in the most relevant existing research works into two main categories; Transformer-based technique such as BERT and hybrid techniques which combines two or more deep neural network (e.g., CNN, RNN, transformers, etc.), to improve the performance of the classification model. All of these techniques are aiming to identify procedures that could remove some of the well-known limitations of working with very long documents.

2.1 Transformer-based

Transformer architecture was first proposed in 2017 by (Vaswani et al., 2017), and it quickly lead to several successful implementations, most notably BERT by (Devlin et al., 2018) and XLNet by (Yang et al., 2019). Those models rely on bidirectional transformations of input that allow for superior tracking of semantic relationships. One of the most important properties of a Transformer such as BERT is that it can be pre-trained and fine-tuned for specific tasks, languages, and subject matters. One major problem with BERT is its limit to sequence length of 510 tokens. Several approaches to this problem were considered, including chunk selection, efficient self-attention, document truncation, and key sentence selection, with one existing method chosen as a representative of each theoretical direction.

(Sun et al., 2019) show how to train and fine-tune BERT for text classification. They tested their model using standardized hyper parameters such as batch size and sequence length, with several different datasets suitable for question classification, topic classification, and sentiment analysis. The best option was determined based on experimental results, for example, in this way it was found that multiple strategies can be used to get around BERT's limit to the sequence length, but the 'head & tails' strategy where only the first 128 and the last 382 tokens are kept performs the best (Pappagari et al., 2019).

On the other hand (Ding et al., 2020) presents a very creative solution to get around BERT's limit to sequence length. This solution inspired by a cognition theory of working memory proposed by Baddeley in 1992. The solution is named CogLTX, and it starts from the logical assumption that a majority of semantically important information is concentrated within specific sentences inside of a longer text, making it unnecessary to check for connections between all words in a document. Instead, they propose training a judge model that can recognize high-relevance sentences and pass them as input to the reasoning model that can complete the classification task.

Another work by (Adhikari et al., 2019) attempt to develop a computationally more efficient version of BERT that would be better suited for classification of long documents. Knowledge distillation is used where knowledge is transferred from a large version of BERT to a much smaller BiLSTM network, which is then used to perform the classification task on new examples. Likewise, (Beltagy et al., 2020) attempt to adjust the successful Transformer architecture and make it better suited for analyzing long documents. Due to exponential increase in computational complexity, with models of this kind it becomes nearly impossible to handle documents whose length exceeds a certain arbitrary threshold. To remedy this issue, the authors propose an altered model they named Longformer, which reduces the complexity of the self-attention matrix.

2.2 Hybrid Methods

Two main approaches that we investigate are algorithms with sparse attention and hierarchical models, each of which has inspired a number of attempts to adjust the DNN architecture for the long document classification tasks. Some works use CNN and utilizing local convolutional feature aggregation to obtain the predictions of the document class (Liu et al., 2018). They proposed a RNN component to the architecture described in the first method, and uses it to track the semantic order for each of the selected sequences. Another work by (He et al., 2019) present a hybrid solution developed specifically with the idea to perform long document classification more efficiently. It combines a RNN-based controller component with a CNN that extracts discriminative features from linguistic content. The controller contextualizes the attention and localizes the extracted bits into a coarse representation of the document, before grouping the extracted features to acquire the overall structure.

Some other ideas are to avoid the exponential growth of complexity along with text sequence length that is typical for architectures of this kind (Huang et al., 2021; Park et al., 2022). For example, (Khandve et al., 2022) propose a hybrid solution with transfer learning as the central principle and a hierarchical architecture that reduces the number of necessary operations. The input data was divided into parts and processed with Universal Sentence Encoder and BERT, both of which were pre-trained. After this, the results were passed onto a shallow network based on either LSTM or CNN concept, which was used in the role of a classifier. Different combinations were explored to determine whether an improvement can be observed, and in case of USE it was possible to improve accuracy in this manner while for BERT the addition of

CNN classifier resulted in similar level of accuracy but with much lower computational requirements thanks to the hierarchical structure.

Another example was developed for the medical domain, and is characterized by strong performance with document of extraordinary length that are typical within this sector (Hu et al., 2021; Si and Roberts, 2021). (Hu et al., 2021) describes a hybrid model that includes several components, including bi-directional recurrent neural network (RNN), convolutional neural network (CNN) and the attention mechanism. Words are embedded as vectors in the initialization phase, before n-grams are extracted from sentences to capture more of the semantic context. A matrix of features is constructed with a combination of CNN output with the ReLu unit.

3 Comparison of the proposed methods

Most of the recent works addressing the problem of long document classification start from similar principles common to all deep learning methods. They also diverge in many aspects, as the authors explore different avenues for leveraging the power of the learning algorithms and overcoming the most significant obstacles (Dai et al., 2022). Since the authors are essentially attempting to solve the same problem, namely how to maintain high accuracy of semantic predictions while keeping the computing demands reasonable, it would be fair to describe the papers as belonging to the same family despite the considerable differences in approach.

To provide an impartial comparison of the proposed models and evaluate the degree of innovation they introduce, it's necessary to look at several elements present in each work, including:

- Basic methodological blueprint used to construct the model
- Priority objectives the authors are trying to achieve
- Statistical operations and training procedures designed to improve accuracy and/or efficiency
- Datasets and standards used for quantitative evaluation of the model
- Potential for real-world applications and follow-up work

In terms of methodological choices, practically all works from this group acknowledge the unmatched power of the attention mechanism for analyzing semantic relationships, and incorporate it in some way into the proposed architecture. There is a division between works that mostly (or completely) embrace an existing architecture and perform only minor operations such as fine-tuning or knowledge transfer in order to reduce the computational demands (Adhikari et al., 2019; Sun et al., 2019). On a different end of the spectrum, there are works that propose innovative hybrid solutions in which the attention mechanism and/or Transformer architecture are combined with elements of different deep learning paradigms, such as RNNs and CNNs. In particular, a common strategy is to adopt a hierarchical structure for the overall solution and use the attention mechanism only in a limited role, thus avoiding the exponential growth of complexity (Huang et al., 2021; Si and Roberts, 2021).

The aforementioned methodological differences stem largely from the expectations for each paper, which range from proving a theoretical point to attempting to develop specialized model for long document classification. Works with a narrower scope tend to stay closer to the original BERT model design (Beltagy et al., 2020), while more ambitious efforts that aim to create new tools are more inclined to experiment with previously untested combinations of elements. In some papers, the scope of intended applications is limited to long documents from a certain domain (i.e. medical) (Si and Roberts, 2021), while others are approaching the problem in more general terms. Finally, there is an important distinction between works that aim for greater accuracy, and those that primarily attempt to improve computational efficiency and shorten the inference time (Park et al., 2022).

It's a fair assessment that practically all works from this group are grappling with the same problem – the tendency of attention-based models to become prohibitively complex as the length of the analyzed text is increased. In response, the authors tried a variety of ideas that rely on vastly different mechanisms to decrease complexity. From fine-tuning and knowledge distillation to introduction of hierarchical architectures and restrictive elements such as fixed-length sliding window (Beltagy et al., 2020; Wang et al., 2020), the proposed techniques are quite innovative and typically leverage some known properties of deep learning models to affect

Table 1: Comparison of the reported performance of different long document classification methods

Author	Method	Reported accuracy
(Liu et al., 2018)	Local convolutional feature aggregation	From 88 to 95.4%
(Sun et al., 2019)	BERT + head and tail truncation	Error rates from 0.67 to 5.4
(He et al., 2019)	Recurrent attention learning	From 77.4 to 80.4%
(Adhikari et al., 2019)	DocBERT	From 54 to 91%
(Pappagari et al., 2019)	RoBERT/ToBERT	From 82 to 95.4%
(Wang et al., 2020)	Dynamic hierarchical topic graph	From 68.8% to 97.3%
(Beltagy et al., 2020)	LongFormer	F1 score from 64.4 to 94.8
(Ding et al., 2020)	CogLTX a BERT-based model + MemRecall algorithm	F1 score from 70% to 97%
(Si and Roberts, 2021)	Hierarchical Transformer	Up to 94.7%
(Huang et al., 2021)	Hybrid self-attention sparse network	From 73.2% to 95.7%
(Khandve et al., 2022)	Hierarchical Attention Network (HAN) + BERT + CNN/LSTM	From 80 to 100%

how the attention mechanism performs in a particular deployment. The diversity of ideas found in those papers illustrates that researchers are currently casting a wide net and searching for unconventional answers to a difficult problem, without a single dominant strategy. On the other hand, hybrid approaches hold a lot of promise and they combine some proven elements from different methodologies into new, potentially more optimal configurations (Huang et al., 2021; He et al., 2019).

Evaluation of the proposed changes to established algorithms is crucially important, and all of the reviewed works include some form of empirical confirmation of their premises. While the numbers seemingly validate that the proposed solutions achieve state-of-the-art results under the best possible conditions, those findings are self-reported and may often be too optimistic. All of the papers are interested in document classification tasks and use it to evaluate their solutions, but datasets used for testing may not be the same in terms of size, diversity, and content. When directly comparing different solutions, it's extremely important to keep in mind the particulars of the evaluation protocols. Studies aiming to provide evaluations with independently administered comparative testing of several different BERT-like algorithms for document classification are slowly emerging and reporting some interesting findings that often diverge from self-assessed results (Wagh et al., 2021; Dai et al., 2022; Park et al., 2022). Still, there are no widely accepted evaluation standards and every comparison suffers from 'apples-to-oranges' problem up to an extent.

When it comes to practical use of the proposed solutions, there is a general lack of field data and even discussions of use cases are rare. This is understandable considering the main focus is on discovering more efficient methods, but without real world testing it's difficult to predict whether any of the solutions can deliver similar results to their reported findings. Some works may be directed as specific niches such as legal (Wan et al., 2019; Bambroo and Awasthi, 2021) or medical (Si and Roberts, 2021), but even in this case little attention is paid to practicalities associated with real world application. This weakness may reflect the current state of the field, which is highly experimental and mostly built on data collected in a controlled environment.

4 Datasets and Reported Accuracy

As previously stated, all papers include an experimental evaluation of the proposed solution and present certain quantitative findings that underscore their methodological choices. In particular, they measure the ability of the model to correctly classify long documents by topic. The performance is typically reported in terms of accuracy with several different metrics, but other aspects may be tracked as well such as complexity or speed of inference. It's important to note that multiple versions of the algorithms are tested on several datasets in each study, which is why accuracy estimations are given as ranges as shown in Table 1.

The choice of the training and testing datasets also carries great significance when analyzing the output of various deep learning algorithms. The same is true for the length of documents, as all of the reviewed papers state among their objectives the improvement of performance with long text sequences. Datasets may also differ by their volume, the number of classes, and other parameters as well, and some studies may include tasks other

Table 2: Overview of the datasets used for training and evaluation with average sequence length

Author	Datasets	Average document length
(Liu et al., 2018)	Custom set comprised of arXiv papers	6000 words
(Wang et al., 2020)	20NG, R8, R52, The Oshumed, MR,	From 39 to 389
(Sun et al., 2019)	IMDB, Yelp reviews, TREC, Yahoo answers, AG News, DBPedia, Sogou	10 – 740 words
(He et al., 2019)	Custom set comprised of arXiv papers	6300 words
(Adhikari et al., 2019)	Reuters, AAPD, IMBD, Yelp 2014	145 -390 words
(Beltagy et al., 2020)	WikiHop, TriviaQA, HotpotQA, OntoNotes, IMDB, Hyperpartisan	from 139 -2000
(Ding et al., 2020)	NewsQA, 20NewsGroups,Alibaba and HotpotQA	from 300-650 Limited by MemRecall block
(Pappagari et al., 2019)	CSAT, 20NG, Fisher Phase I	260 – 1790 words
(Si and Roberts, 2021)	MIMIC III	700 tokens
(Huang et al., 2021)	IMDB, Yelp 2018	From 100 to 500 words per review
(Khandve et al., 2022)	20NG, BBC News, AG News, BBC Sports, IMDB, R8	From 39 to 389 words per document

than document classification. The overall datasets used for training and evaluation with average sequence length are presented in Table 2.

5 Conclusion and Future Directions

It is beyond doubt that Transformer architecture changed the way linguistic analysis is performed, and in a very short time BERT has been widely accepted as the golden standard of semantic understanding. However, the greatest value of this concept may be tied to its flexibility, as it allows for extensive customization and specialization with only minimal modifications of the training procedure. While there have been numerous adaptations of successful Transformer models in the past, it's highly likely that the number and quality of derivative work will increase in the near future. Figuring out ways to improve an already impressive model is not easy, but growing presence of this topic in the online forums and greater availability of research papers dealing with some of the outstanding challenges could power the next wave of research in this direction. This process is already underway, and a breakthrough achieved with Transformers is being actively exploited by research teams from around the world.

Computational efficiency remains the central challenge, and developing models that can achieve elite accuracy on a wide range of tasks without requiring escalating amount of resources is a top priority for the next stage of research. Some of the ideas presented in the reviewed works will certainly be revisited and expanded in the coming years, and their cumulative contributions could eventually lead to a consensus solution. In parallel with the process of consolidation of knowledge and resolving practical difficulties, we can also expect to see a larger number of domain-specific applications that are designed and trained with real-world use in mind. Since in many domains there are long documents to be classified, solving the difficulties that current algorithms are having with long text sequences will stay a key objective. Localization is another issue that should be addressed in future work, as most of the current tools were never tested with non-English datasets. Given that the volume of non-English documents is enormous and growing very fast, it would be refreshing to see language-specific applications that match the quality of original BERT.

Hybridization of models remains an area that hasn't been sufficiently explored, in part due to huge potential for mixing and matching different elements. The advantages offered by older paradigms such as recurrent or convolutional neural networks shouldn't be ignored, and some very imaginative efforts to combine them with the attention mechanism were made. Hybrid approaches to long document classification are rapidly emerging over the last few years (Qin et al., 2022), and some of them deserve to be explored further. Balancing complexity of the model and compatibility of all components presents a unique challenge, and it may take several years before fully mature solutions of this type start appearing.

6 Acknowledgments

Author wants to express their gratitude towards the Research Department in Elm Company for funding and support this project.

References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*.

Purbid Bambroo and Aditi Awasthi. 2021. Legaldb: Long distilbert for legal document classification. In

2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), pages 1–4. IEEE.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Xiang Dai, Ilias Chalkidis, Sune Darkner, and Desmond Elliott. 2022. Revisiting transformer-based models for long document classification. *arXiv preprint arXiv:2204.06683*.

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

Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Cogltx: Applying bert to long texts. *Advances in Neural Information Processing Systems*, 33:12792–12804.

Jun He, Liqun Wang, Liu Liu, Jiao Feng, and Hao Wu. 2019. Long document classification from local word glimpses via recurrent attention learning. *IEEE Access*, 7:40707–40718.

Yongli Hu, Puman Chen, Tengfei Liu, Junbin Gao, Yanfeng Sun, and Baocai Yin. 2021. Hierarchical attention transformer networks for long document classification. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE.

Weichun Huang, Ziqiang Tao, Xiaohui Huang, Liyan Xiong, and Jia Yu. 2021. Hierarchical self-attention hybrid sparse networks for document classification. *Mathematical Problems in Engineering*, 2021.

Snehal Ishwar Khandve, Vedangi Kishor Wagh, Apurva Dinesh Wani, Isha Mandar Joshi, and Raviraj Bhuminand Joshi. 2022. Hierarchical neural network approaches for long document classification. In *2022 14th International Conference on Machine Learning and Computing (ICMLC)*, pages 115–119.

Liu Liu, Kaile Liu, Zhenghai Cong, Jiali Zhao, Yefei Ji, and Jun He. 2018. Long length document classification by local convolutional feature aggregation. *Algorithms*, 11(8):109.

Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844. IEEE.

Hyunji Hayley Park, Yogarshi Vyas, and Kashif Shah. 2022. Efficient classification of long documents using transformers. *arXiv preprint arXiv:2203.11258*.

Ruyu Qin, Min Huang, Jiawei Liu, and Qinghai Miao. 2022. Hybrid attention-based transformer for long-range document classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Yuqi Si and Kirk Roberts. 2021. Hierarchical transformer networks for longitudinal clinical document classification. *arXiv preprint arXiv:2104.08444*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China national conference on Chinese computational linguistics*, pages 194–206. Springer.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*.

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

Vedangi Wagh, Snehal Khandve, Isha Joshi, Apurva Wani, Geetanjali Kale, and Raviraj Joshi. 2021. Comparative study of long document classification. In *TENCON 2021-2021 IEEE Region 10 Conference (TENCON)*, pages 732–737. IEEE.

Lulu Wan, George Papageorgiou, Michael Seddon, and Mirko Bernardoni. 2019. Long-length legal document classification. *arXiv preprint arXiv:1912.06905*.

Zhengjue Wang, Chaojie Wang, Hao Zhang, Zhibin Duan, Mingyuan Zhou, and Bo Chen. 2020. Learning dynamic hierarchical topic graph with graph convolutional network for document classification. In *International Conference on Artificial Intelligence and Statistics*, pages 3959–3969. PMLR.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Optimizing singular value based similarity measures for document similarity comparisons

Jarkko Lagus
IPRally Technologies Oy and
University of Helsinki
jarkko@iprally.com

Arto Klami
Department of Computer Science
University of Helsinki
arto.klami@helsinki.fi

Abstract

The similarity of documents is typically computed using fairly simple similarity measures, such as mean or maximum pooling of word representations followed by vector cosine similarity. This results in fast computation but compared to second-order or matrix-based similarity measures loses information. In this work, we investigate the value of matrix similarity measures for document similarity comparison in full-length patent retrieval tasks and introduce two new metrics motivated by the Schatten p-norm. The new similarity measures are based on singular values and involve learnable parameters to be optimized for a given evaluation task. We show that tuning the similarity measures for a specific task improves the similarity comparison accuracy.

1 Introduction

1.1 Document representations and similarity

For natural language processing tasks, we typically represent words and documents as numerical vectors, since they allow mathematically simple comparisons (e.g. similarity between two documents) and are space-efficient. Modern vector representation methods are highly informative for individual words and even long documents can be represented as relatively low-dimensional vectors. Already simple mean pooling can work well in practice (Conneau et al., 2018), and further developments such as smart weighting schemes (Arora et al., 2017; Gupta et al., 2020), directly learned document vector representations (Le and Mikolov, 2014; Chen, 2017), and especially the contextual embeddings and transformer models (Vaswani et al., 2017; Devlin et al., 2018) have pushed the limits of what one can encode into a vector. Transformers, however, have often very high computational cost (Sharir et al., 2020) and simpler methods and better similarity measures based on static word representations still have their place in many applications.

We step outside such vector-shaped representations and directly work with a full matrix $A \in R^{n \times d}$ that stores d-dimensional representations for n words appearing in the document. We explore the value of covariance pooling and singular value (SV) based similarity measures in patent similarity comparison tasks, and show that in the case of static embeddings, these similarity measures outperform mean vector representations in full document comparison tasks.

The key contribution of this work is the introduction of new matrix similarity measures for document similarity. We explain how submultiplicative norms can be converted into a metric resembling cosine similarity, providing a family of similarity measures building on the Schatten p-norm (later just p-norm) computed using SVs of covariance pooling. We then introduce new similarity measures that are based on the same SVs but map them to similarity scores in a more flexible manner. The new similarity measures have learnable parameters that are tuned for a specific end task and hence can learn to represent relevant information better.

1.2 Matrix metrics

We define a matrix similarity measure to be a function $f(A, B) \in R$ that assigns similarity score for document matrices $A \in R^{n \times d}$ and $B \in R^{m \times d}$, where d is the dimensionality of the word embedding vectors (here $d = 300$), n is the amount of tokens in the document A, and m is the amount of tokens in document B.

The similarity between matrices can be defined in multiple ways. The most straightforward ones – such as Word mover's distance (Kusner et al., 2015) (also known as Bures-Wasserstein distance (Bhatia et al., 2019)) or pairwise comparison of all possible word pairs in a matrix – can be directly applied on matrices with an arbitrary number of rows, and hence for documents of arbitrary lengths. Some other similarity measures assume A and B

to be the same shape. To apply those for document comparisons, we need to first preprocess the document matrices suitably; we here call this step *pooling*. The simplest pooling approach is *padding* the shorter document with suitably many rows of zeros, whereas a more general approach is to use *covariance pooling* where we use $A^TA \in R^{d\times d}$ and $B^TB \in R^{d\times d}$ as the inputs for the similarity measure. Covariance pooling has been shown to have beneficial properties as a document representation (Torki, 2018; Lagus et al., 2019). As d is often large, for the smaller size we can use *SVD pooling* where only k leading singular vectors of the covariance representation are used. This can have a regularizing effect in addition to lowering memory and computational costs (Lagus et al., 2019).

1.3 Patent retrieval as context

We evaluate the measures in the context of patent applications, as an example domain with long but structured documents. Tools for handling patent documents are in high demand due to the high labor cost of manual inspection. This is especially the case for the invalidity search stage, aiming to find relevant patents that could possibly cause issues with e.g. patent infringement, or lead to delays or rejection of the application. Over the years, there has been lots of research on how to automate different parts of the process (Balsmeier et al., 2018; Aristodemou and Tietze, 2018) and on end-to-end solutions (Gao et al., 2022) for specific tasks. In addition to trying to solve specific tasks, there have been efforts toward creating patent-text-specific language models (Lee and Hsiang, 2020; Bekamiri et al., 2021). The patent domain is ideal for exploring alternative similarity measures as the documents are often tens of pages long and better methods are needed to use the full information.

2 New similarity measures

This section introduces our technical contributions. We first explain how submultiplicative matrix norms can be used for deriving a similarity measure between two matrices and provide a family of measures building on the p-norm, computed using SVs of the covariance pooling of document matrices. We then introduce a family of more expressive matrix similarity measures, replacing the matrix norm with alternative functions of the SVs. The new measures have learnable parameters that can be fine-tuned for a given task.

2.1 From matrix norm to similarity measure

Any submultiplicative matrix norm $\|A\|$ satisfying $\|AB\| \leq \|A\|\|B\|$ can be used as a basis for a normalized similarity measure between matrices A and B. If we denote the norm (or norm-like function) with $S(\cdot)$, we get the general formula

$$D(A, B, S(\cdot)) := \frac{S(A^TB)}{S(A^TA)^{1/2}S(B^TB)^{1/2}}. \quad (1)$$

This measure is a natural extension to the standard cosine similarity between vectors. Due to submultiplicativity, it is always within the range $[-1, 1]$. Even though the measure will not in general be a proper metric, we will have higher similarity when A and B are similar in terms of the norm and can use it for similarity comparisons.

We build on a particular family of submultiplicative norms called Schatten p-norms, defined as

$$S_p(A) := \left(\sum_n s_n^p(A)\right)^{1/p}, \quad (2)$$

where $p \in [1, \infty)$ and $s_n(A)$ is the nth SV of the matrix A in descending order. The normalized similarity measure can then be expressed as $D(A, B, S_p(\cdot))$ in the general notation of Eq. (1). This family generalizes several well-known norms: for $p = 2$ we get the Frobenius norm, for $p = 1$ it corresponds to the trace norm, and for $p = \infty$ we get the operator norm. Lagus et al. (2019) presented the similarity measure of Eq. (1) in the specific context of the Frobenius form, but here we consider the general formulation for arbitrary norms and norm-like functions.

For $p \in (0, 1)$ the p-norm is a quasinorm as it does not fulfill the triangle inequality, but we still have $D(A, B, S_p(\cdot)) \in [-1, 1]$ and hence get a normalized similarity measure. The p-quasinorm has gained traction in other matrix applications such as low-rank matrix recovery (Zhang et al., 2019) and image denoising (Xie et al., 2016).

2.2 Learnable similarity measures

The measure (1) depends on the norm. Instead of assuming a specific norm in advance, we propose using a slightly more flexible parametric family of norms. We can then optimize the parameters of the norm directly for a task where the distance measure is used. The p-norm (2) itself has the parameter p which can be learned to maximize a task performance, such as retrieval accuracy.

For more flexibility, we propose extensions of the p-norm that involve additional control parameters. We start from the observation that the p-norm is based on SVs, and construct two alternatives that use SVs as inputs.

The simplest extension

$$S_{w,p}(A) := \left(\sum_n \left(w_n s_n(A) \right)^p \right)^{1/p} \quad (3)$$

weights each SV independently but otherwise retains the functional form of the p-norm. This generalization is still a norm, since for any matrix A, we can always find matrix A' where $s_i(A') = w_i s_i(A)$. One motivation for this norm is the observation of Arora et al. (2017) that removing the direction of the largest singular vector reduces the effect of the most common words that are often uninformative. For $p = 1$ (denoted as $S_{w,1}(\cdot)$ later on) we obtain simple weighting as special case of the more general weighting. Alternatively, we can interpret the weights w_n as a form of an attention mechanism.

As a still more flexible alternative, we consider directly mapping the SVs of $A^T B$ to the similarity with a flexible model. We can then include the normalization within the measure itself, and hence get directly a replacement for Eq. (1). For this, we use a small neural network

$$\mathrm{D}_{NN}(A, B) = T(R(R(s(A^T B)W_1)W_2)W_3),$$

where $R(\cdot)$ is a the rectified linear unit activation function and the layer weights

$$W_1 \in R^{d \times 500},$$
$$W_2 \in R^{500 \times 500}, \text{and}$$
$$W_3 \in R^{500 \times 1}.$$

Finally, the hyperbolic tangent $T(\cdot)$ at the end ensures the output is normalized between $[-1, 1]$. Each layer has also a bias term of suitable size, which is omitted here for conciseness. The network architecture could be further tuned by standard architecture search and hence this architecture is to be seen as one practical example of the more general approach.

3 Experiments

We evaluate the proposed similarity measures in the context of patents. When patent examiners evaluate the novelty of a patent application, there are different kind of prior art that is to be considered. The *X citations* are prior work that can alone lead to a rejection, while the *A citations* describe the state of the art, but are not immediate reasons for rejection. Differentiating between these categories can be useful, for example, in retrieval tasks where we want to rank the patents by their relevance to the original document. If we know the relative ordering of each citation class, we can reorder the search results to highlight the most relevant documents, i.e. in this case *X*s before *A*s.

Patents themselves consist of two main parts, *Claims* and *Description*, where the Claims part describes the actual claims that are being made and the description part is a more free-form description of the invention overall. For this reason, the Claims part is usually much shorter and less noisy than the Description part, while the Description part is more thorough and thus contains more fine-grained information. We evaluate the similarity measures for both cases to provide two parallel sets of results.

3.1 Data and evaluation

Documents and encoding The dataset consist of 3,500 full-length patent applications acquired from the United States Patent and Trademark Office, with average document length being 37,754 characters for the Descriptions and 1,907 characters for the Claims. We encode the patent documents using English 300-dimensional `fastText` embeddings (Joulin et al., 2016) and form the covariance matrices of dimensionality 300×300 of each document as the representation.

Training For the models that require learning the parameters, we use `PyTorch` (Paszke et al., 2019) library to do gradient-based optimization using 2,000 samples as the training set and 500 samples as the validation set. We use triplet loss as the loss function setting one of the models as the distance function and the margin (chosen using hyperparameter optimization) to 0.5. The loss for one instance for the measure in Eq. (1) is then

$$\max(D(A, P, S(\cdot)) - D(A, N, S(\cdot)) + 0.5, 0),$$

and for the neural network model it is

$$\max(\mathrm{D}_{NN}(A, P) - \mathrm{D}_{NN}(A, N) + 0.5, 0),$$

where A is the encoded original document, P is the encoded X citation (positive sample), N is the encoded A citation (negative sample), and $S(\cdot)$ is a norm-like measure. Optimization is terminated once the result on the validation set decreases for three consecutive evaluations.

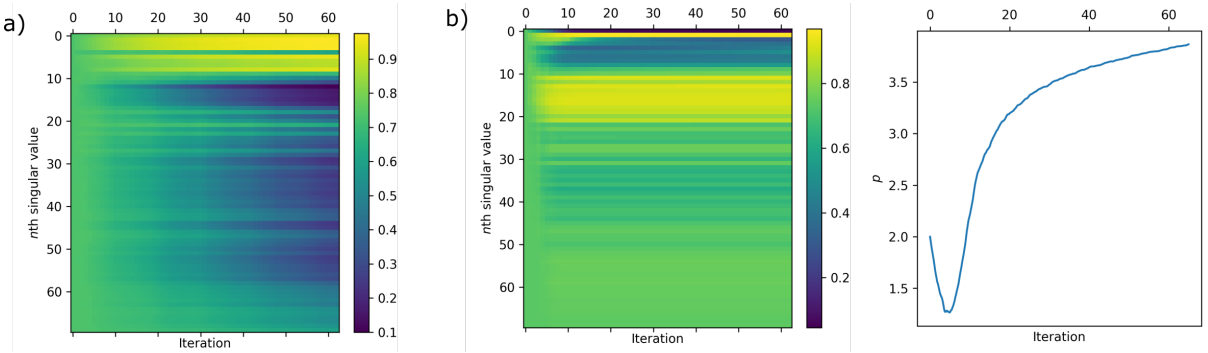


Figure 1: a) Development of singular value weights as a function of iterations for the model $S_{w,1}$. b) Development of the weights and p for the model $S_{w,p}$. Only first 70 out of 300 weights are shown; the rest are effectively zero.

Dataset	Mean	$S_{0.1}$	$S_{0.2}$	$S_{0.5}$	$S_{1.0}$	$S_{1.5}$	$S_{2.0}$	$S_{5.0}$	S_{∞}	S_{opt}	$S_{w,1}$	$S_{w,p}$	D_{NN}
Claims	0.566	0.593	0.601	0.580	0.594	0.577	0.558	0.545	0.545	0.603	0.588	0.589	0.642
Descr.	0.553	0.549	0.558	0.573	0.520	0.504	0.496	0.482	0.482	0.574	0.525	0.574	0.652

Table 1: Numerical results. *Mean* shows the baseline of mean vector with cosine similarity. Free-form neural network model D_{NN} is clearly the best for both tasks.

Evaluation Finally we evaluate the trained model using a test set of 1,000 triplets, measuring the distance from the anchor to both positive and negative samples and counting how often the positive sample is closer to the anchor than the negative sample, i.e. the X citation ranks higher than the A citation. As the baseline, we use the standard mean vector combined with cosine similarity.

3.2 Results

Results are reported in Table 1. We first inspect the accuracy using standard p-norm by grid search over p. The main observation is that small values of p are the best, so that $p = 1$ is the best of the proper norms in both cases and the highest overall accuracy is obtained with quasinorms with $p < 1$. The best p clearly outperforms the baseline of mean vector and cosine similarity (*Mean*); for Claims we improve from 0.566 to 0.601 with $p = 0.2$ and for Descriptions from 0.553 to 0.573 with $p = 0.5$. Large p are clearly worse and all $p > 3$ are effectively equivalent to $p = \infty$.

Instead of evaluating the metric for a range of p, we can directly optimize over p. For both cases, the solution (S_{opt}), slightly improves from the one chosen amongst the grid of alternatives as expected, with optimal values of $p = 0.884$ for Claims and $p = 0.327$ for Descriptions. One technical aspect we note is that when $p \in (0, 1)$ the function is non-convex (Shang et al., 2020) and can have multiple local optima within this range, but we did not observe this to be a problem in practice.

The weighted extension of p-norm of (3) is denoted here by $S_{w,p}$. Figure 1 (a) illustrates the learned weights (as function of iteration) for fixed $p = 1$, demonstrating how the measure assigns more weight for the first 10 or so SVs. Figure 1 (b) illustrates the behavior of the weights and p when optimized jointly, and reveals quite different phenomena: Instead of small p it is now better to use large p and down-weight many of the early singular vectors. For both Claims and Descriptions, the weighted variant $S_{w,p}$ outperforms the mean baseline, but does not provide an improvement over S_{opt} and for Claims it remains worse. One advantage of these measures is that – as seen here – the similarity measures only depend on a fairly small number of SVs; it is enough to compute some tens of the SVs rather than all 300.

The still more flexible neural network measure D_{NN} works well, reaching the highest accuracy for both Claims and Descriptions, with substantial improvement also over S_{opt}. This verifies that SVs of $A^T B$ can be used as the basis for accurately measuring similarity between documents. Importantly, we have high accuracy also for the full-length documents (Descriptions) that are challenging for all other similarity measures.

4 Conclusions

We set out to investigate how similarity measures based on matrix norms work in document similarity comparisons in the context of patent retrieval. We focused on similarity measures based on singular

values of the inner product of the two document matrices, motivated by the p-norm. Our main contribution was introducing new parametric similarity measures that build on the same singular values but are fine-tuned for the specific task at hand, and we showed how a direct neural network mapping the singular values to a distance outperforms both standard mean representation as well as our attempts of more constrained and interpretable measures. In this work we did not fine-tune the neural network architecture to maximize the accuracy but rather used a generic small network, but for practical use the network architecture could be tuned to further improve the accuracy.

While the work was done in the context of static embeddings and patent data, the applicability is not limited to these. Likely any full-document comparison task can benefit from richer representations and the contextual embedding models should enhance the results even further.

Acknowledgments

This work was supported by the Academy of Finland Flagship programme: Finnish Center for Artificial Intelligence FCAI. We thank Janne Sinkkonen for the initial idea and insightful discussions related to the use of Schatten p norm in document comparison.

References

Leonidas Aristodemou and Frank Tietze. 2018. The state-of-the-art on intellectual property analytics (ipa): A literature review on artificial intelligence, machine learning and deep learning methods for analysing intellectual property (ip) data. *World Patent Information*, 55:37–51.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*.

Benjamin Balsmeier, Mohamad Assaf, Tyler Chesebro, Gabe Fierro, Kevin Johnson, Scott Johnson, Guan-Cheng Li, Sonja Lück, Doug O'Reagan, Bill Yeh, et al. 2018. Machine learning and natural language processing on the patent corpus: Data, tools, and new measures. *Journal of Economics & Management Strategy*, 27(3):535–553.

Hamid Bekamiri, Daniel S Hain, and Roman Jurowetzki. 2021. Patentsberta: A deep nlp based hybrid model for patent distance and classification using augmented sbert. *arXiv preprint arXiv:2103.11933*.

Rajendra Bhatia, Tanvi Jain, and Yongdo Lim. 2019. On the bures–wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 37(2):165–191.

Minmin Chen. 2017. Efficient vector representation for documents through corruption. *arXiv preprint arXiv:1707.02377*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*.

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

Xiaochen Gao, Zhaoyi Hou, Yifei Ning, Kewen Zhao, Beilei He, Jingbo Shang, and Vish Krishnan. 2022. Towards comprehensive patent approval predictions: Beyond traditional document classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 349–372.

Vivek Gupta, Ankit Saw, Pegah Nokhiz, Praneeth Netrapalli, Piyush Rai, and Partha Talukdar. 2020. P-sif: Document embeddings using partition averaging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7863–7870.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR.

Jarkko Lagus, Janne Sinkkonen, Arto Klami, et al. 2019. Low-rank approximations of second-order document representations. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. ACL.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.

Jieh-Sheng Lee and Jieh Hsiang. 2020. Patent classification by fine-tuning bert language model. *World Patent Information*, 61:101965.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch:

An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Fanhua Shang, Yuanyuan Liu, Fanjie Shang, Hongying Liu, Lin Kong, and Licheng Jiao. 2020. A unified scalable equivalent formulation for schatten quasi-norms. *Mathematics*, 8(8):1325.

Or Sharir, Barak Peleg, and Yoav Shoham. 2020. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*.

Marwan Torki. 2018. A document descriptor using covariance of word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 527–532.

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

Yuan Xie, Shuhang Gu, Yan Liu, Wangmeng Zuo, Wensheng Zhang, and Lei Zhang. 2016. Weighted schatten p-norm minimization for image denoising and background subtraction. *IEEE transactions on image processing*, 25(10):4842–4857.

Hengmin Zhang, Jianjun Qian, Bob Zhang, Jian Yang, Chen Gong, and Yang Wei. 2019. Low-rank matrix recovery via modified schatten-p norm minimization with convergence guarantees. *IEEE Transactions on Image Processing*, 29:3132–3142.

Semantic Similarity Based Filtering for Turkish Paraphrase Dataset Creation

Besher Alkurdi and **Hasan Yunus Sarioglu** and **Mehmet Fatih Amasyali**
Department of Computer Engineering
Yildiz Technical Univeristy
Istanbul, Turkey
{besher.alkurdi, yunus.sarioglu}@std.yildiz.edu.tr
amasyali@yildiz.edu.tr

Abstract

In this study, we introduce a new method for creating paraphrase datasets from parallel bilingual corpora. We also introduce large paraphrase datasets created using this method. We utilize machine translation to create paraphrase datasets by translating the English phrases in Turkish-English parallel datasets to Turkish. Detailed pre-processing steps are applied to the text pairs. A sample from our translated datasets was annotated by native speakers for semantic similarity, and a model with the same task was chosen based on the correlation with human annotations. We then filtered the pre-processed and translated text pairs by semantic similarity calculated by the chosen model. Two pre-trained encoder-decoder architectures were fine-tuned on the datasets that we created. We present results asserting our data collection and filtering method's effectiveness.

1 Introduction

Paraphrase generation can be applied in several fields including data augmentation (Kumar et al., 2019), machine translation evaluation (Thompson and Post, 2020), chatbots (Garg et al., 2021), question answering (Zhu et al., 2017), and semantic parsing (Cao et al., 2020). A major challenge in paraphrase generation research is the lack of large paraphrase datasets, especially in languages other than English. This served as a motivation for us to create high-quality and large paraphrase datasets in Turkish. We use English-Turkish datasets and translate the sentences from English to Turkish. Semantic similarity is then calculated using a Transformer-based (Vaswani et al., 2017) model for each pair in the resulting Turkish-Turkish datasets. Pairs that have a score greater than a threshold are accepted as paraphrases. The threshold is chosen in accordance to human annotations collected by us.

Our main contributions are as follows:

- We present the largest Turkish paraphrase datasets yet consisting of approximately 800k pairs in total.
- We introduce a new method for creating a paraphrase dataset from a parallel corpus combining machine translation and semantic similarity based filtering.
- We share paraphrase generation models trained on the datasets we introduce as part of our work and evaluate them using several benchmark metrics.
- We share a manually annotated semantic textual similarity dataset consisting of 500 pairs.

The datasets and the fine-tuned models are shared publicly.[1] We hope that our work encourages more research in this area, and provides a dataset that can be used for benchmarking paraphrase generation architectures and datasets in the future.

2 Related Work

The task of finding texts with similar or identical meaning, often called paraphrase identification is a challenging task. Several approaches have been tried to create paraphrase datasets in previous work.

Manual paraphrase collection is very expensive, unscalable and implausible with limited resources. Studies in this area have usually made use of crowd sourcing to construct a paraphrase dataset (Burrows et al., 2013). The main advantage of this method is its effectiveness in constructing a high-quality dataset where diversity of the sentences can be increased without the fear of producing pairs with low semantic similarity.

Semantic similarity based mining can be employed to detect paraphrases in a corpus of texts. Each sentence is compared with every other sentence in the corpus and given a similarity score, the

[1] https://github.com/mrbesher/semantic-filtering-for-paraphrasing

sentence with the highest score is considered a paraphrase. This method suffers from quadratic runtime and thus fails to scale to large paraphrase datasets. A similar approach was employed in (Martin et al., 2020).

Machine translation can be used where a text is translated to a pivot language then to the source language again (Wieting and Gimpel, 2018), (Wieting and Gimpel, 2018), (Suzuki et al., 2017). Multiple pivot languages can be used in a similar manner. While this method is successful, it may suffer from noise caused by automatic translation from the source to the pivot language and back from it.

Other automatic approaches were used like using parallel movie subtitles (Aulamo et al., 2020), image captions of the same image (Lin et al., 2014), and texts that can be marked as paraphrases based on different conditions such as duplicate questions,[2] duplicate posts (Lan et al., 2017), and text rewritings (Max and Wisniewski, 2010).

A handful of research on Turkish paraphrase dataset creation have been shared. (Karaoğlan et al., 2016) conduct a study resulting in 2,472 text pairs annotated by humans. (Demir et al., 2013) present a paraphrase dataset consisting of 1,270 paraphrase pairs from different sources. The mentioned datasets are not shared publicly. (Bağcı and Amasyali, 2021) present a combination of translated and manually generated datasets focusing on question pairs, and train a BERT2BERT architecture on it. None of the existing studies provide a comprehensive dataset in Turkish to the best of our knowledge.

3 Dataset Creation

The dataset creation process pipeline consists of several steps to ensure that the dataset is of high quality. Firstly, English-Turkish parallel texts with only one source and one target were downloaded using Opus Tools (Aulamo et al., 2020).

We considered using the datasets shared on OPUS (Tiedemann, 2012).[3] The following datasets where downloaded, examined, filtered and machine translated:

- **OpenSubtitles2018:** A large database of movie and TV subtitles across 60 languages[4] compiled, pre-processed and aligned by (Lison and Tiedemann, 2016).

[2]https://www.kaggle.com/c/quora-question-pairs
[3]https://opus.nlpl.eu/
[4]http://www.opensubtitles.org/

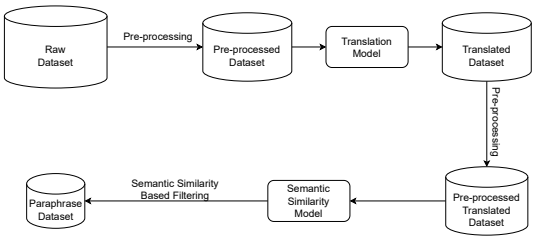


Figure 1: Dataset Creation Pipeline

- **TED2013:** A parallel dataset of TED talk subtitles originally provided by Web Inventory of Transcribed and Translated Talks.[5] The talks were translated automatically, leading to significant noise.

- **Tatoeba v2022-03-03:** A collaborative collection of sentences and translations, compiled using crowdsourcing.[6]

Text pairs were pre-processed according to the characteristics of each dataset to remove unwanted text pairs. An example is removing the explanations done in the TED dataset indicated by two hyphens before and after the explanation while keeping text in square brackets as they made the statements more understandable.

Machine translation is applied on the whole dataset from English to Turkish. At this stage the dataset contains valid text pairs that are candidates to be paraphrases. Source and translated sentences were removed if one includes the other and pre-processing steps were applied again to remove noisy texts generated by the translation model. After that, semantic similarity between text pairs is measured and pairs with a high semantic similarity score are chosen as paraphrases. The steps, illustrated in Figure 1, ensure a robust process to create a high-quality paraphrase dataset.

4 Translation and Semantic Similarity Based Filtering

4.1 Translation

Due to the huge volume of data that we aimed to translate, usage of online machine translation services was unfeasible due to restrictions set by the providers. We chose a machine translation model provided by OPUS-MT project (Tiedemann and

[5]https://wit3.fbk.eu/
[6]https://tatoeba.org

Thottingal, 2020) and shared publicly on Hugging Face.[7]

4.2 Human Annotations for Ground Truth Semantic Similarity

To filter the pairs further, we considered using a semantic similarity metric to remove pairs with low semantic similarity. We had several models to achieve the task of semantic similarity scoring to choose from. For model selection, we sampled 250, 150, and 100 pairs from OpenSubtitles2018, Tatoeba, and TED2013 respectively. The samples were then annotated by 6 native Turkish speakers, with each pair assigned to two different annotators. Following (Creutz, 2018), each pair could be assigned one of the labels described in Table 1. If the annotators disagreed and the score difference was less than two, the label indicating less semantic similarity was chosen. Otherwise, the label was discarded.

A bot was created on Telegram[8] to ease the process of annotation collection and the scores collected from annotators were used later to determine a threshold to filter out low quality paraphrases.

Annotators disagreed by two points on 16 samples from OpenSubtitles2018 (OST), 5 samples from TED2013 (TED), and 3 samples Tatoeba (TAT). Therefore, a total of 24 samples were removed. The distribution of the labels in each dataset is shown in Table 2.

The desired phrase pairs are the ones labeled as near-synonyms or synonyms. 66.32%, 70.94% and 88.44% of the pairs in TED, OST and TAT respectively can be considered paraphrases accordingly. The results show a need for further filtering as phrases with different meanings are expected to affect the model's performance.

4.3 Semantic Similarity Based Filtering

Several semantic similarity models were considered to filter the text pairs. The goal is to capture the closeness in meaning between two input texts. The models we considered utilize BERT as a baseline (Devlin et al., 2019). Among those are Bi-encoders, two identical encoders that compute embeddings of sentences separately. Cosine similarity is then calculated between the embedding pair. We considered three pretrained models of this kind:

- **distiluse-base-multilingual-cased:** Presented in (Reimers and Gurevych, 2020), this model creates multilingual sentence embeddings. The training objective is to map translated sentences' embeddings to the embeddings of the original sentences.

- **multilingual-l12:** This model maps texts to a 384 dimensional dense vector space, the model is shared on Huggingface.[9]

- **emrecan:** This model was fine-tuned on a machine translated version of STS-b[10] and NLI (Budur et al., 2020) to map texts to a 768 dimensional dense vector space. Contrary to the models mentioned before, this model is trained on Turkish datasets.

Cross-encoder networks (Reimers and Gurevych, 2019) accept sentence pairs as inputs, and output the semantic similarity between sentences. Following (Beken Fikri et al., 2021), and using their STS-b (Cer et al., 2017) dataset, which was translated by the authors using Google Cloud Translation API.[11] We fine-tuned BERTurk[12] starting from 5 random seeds for 4 epochs and used the model with the highest correlation score with the similarity labels on the development set split provided by the authors.

We chose the semantic similarity model that filtered out the least amount of pairs labeled as synonyms or near-synonyms. The goal is to remove pairs labeled as having distant meanings or no relevance. We chose thresholds for each model such that after filtering out pairs below the thresholds in the sample annotated by humans, 95% of the kept pairs are labeled as synonyms or near-synonyms. The percentage of the valid pairs kept can be seen in Table 3 for every model. Emrecan was chosen for filtering due to its superiority to the other models.

Table 4, shows the number of text pairs before any filtering was applied in the raw column and the number of kept pairs after pre-processing, prior to translation. The number of pairs kept after semantic similarity based filtering is shown in the last column.

[7] https://huggingface.co/Helsinki-NLP/opus-tatoeba-en-tr
[8] https://telegram.org/
[9] https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2
[10] https://huggingface.co/datasets/emrecan/stsb-mt-turkish
[11] https://github.com/verimsu/stsb-tr
[12] https://huggingface.co/dbmdz/bert-base-turkish-cased

Category	Description	Example
Eş Anlamlı *Synonyms*	İki cümle birbirlerinin yerine kullanılabilir ve temelde aynı anlama gelmektedir. *The two sentences can be used interchangeably and essentially mean the same thing*	Ona yaklaşmayın, hasta olabilir. Ondan uzak durun! Hasta olma ihtimali var. *Do not get close to him. He might be sick.* *Stay away from him! There is a chance that he is sick.*
Yakın Anlamlı *Near-synonyms*	Cümlelerin tarzları farklı olsa da iki cümlenin aynı anlama geldiğini düşünmek mümkün. *Even though the style of the sentences is different they can be thought to have the same meaning.*	O, saçını yapma tarzını değiştirdi. Saçının şeklini değiştirmiş. *She changed the way she does her hair.* *She changed the shape of her hair.*
Uzak Anlamlı *Distant Meanings*	İki cümlenin neden yan yana geldiği anlaşılabilir ancak aynı anlama geldikleri söylenemez. *It can be explained why the sentences are coupled together but one cannot consider them to have the same meaning*	Farklı roller için de seçmelere katılmıştım Birkaç rol için bekledim. *I attended the auditions for different roles.* *I waited for some roles.*
Alakaları Yok *No Relevance*	Cümleler arasında bir bağlantı yok. Farklı anlamlara sahipler. *The sentences have no connection. They have different meanings.*	Afedersin bana benim iki elim yeter. Üzgünüm, sadece ikisini alabilirim. *Execuse me, my two hands are enough for me.* *Sorry, I can only take two of them.*

Table 1: Semantic Similarity Labeling Task Description for Human Annotators

Label	OST	TAT	TED
No Relevance	25	2	6
Distant Meanings	43	15	26
Near-synonyms	92	40	37
Synonyms	74	90	26

Table 2: The distribution of human annotations across the datasets

Model	OST	TAT	TED
BERTurk	33.73	**33.85**	42.86
Distiluse	40.36	8.46	34.92
Multilingual-l12	36.75	9.23	36.50
Emrecan	**42.68**	26.92	**46.03**

Table 3: Percentage of the Kept Valid Pairs

5 Experiments

We ran experiments to measure the quality of our constructed datasets. These are intended to be used as a baseline for future research on Turkish paraphrase generation. We train our models on the unfiltered and the filtered versions of our datasets to analyze the applied filtering method's impact on the quality of our datasets.

For our experiments, we randomly select 5% of the pairs in each dataset as development split and 5% as test split. The rest of the pairs are used for training the models. In this section we present the experimental results of the models we fine-tuned on the train splits and tested on the test splits of our datasets. We employ transfer learning using pre-trained Text-to-Text Transformer models. mT5 is a multilingual variant of T5 presented in (Xue et al., 2021). We use a pre-trained checkpoint of mT5-base provided by Google and published on Hugging Face.[13] We also utilized BART (Lewis et al., 2020) using trBART, a checkpoint of BART-base (uncased) pre-trained from scratch by (Safaya et al., 2022). The authors published the model on Hugging Face.[14]

In our initial experiments, models fine-tuned

[13]https://huggingface.co/google/mt5-base

[14]https://huggingface.co/mukayese/bart-base-turkish-sum

Name	Raw	Pre-processing	Similarity Based Filtering
OST	13,190,557	1,944,955	706,468
TAT	393,876	265,203	50,423
TED	131,874	104,238	39,763

Table 4: Number of Text Pairs in the Datasets Before and After Filtering

on the TED dataset failed to generate acceptable parahrases. We did not continue experimenting with the dataset, and thus only provide the translations and the filtered dataset without experiment results.

Our models were trained for 4 epochs with a learning rate of $1e-4$ on the OST dataset, and for 6 epochs with a learning rate of $1e-4$ on the TAT dataset. Those values yielded the highest BLEU scores of the models on the development splits after several experiments with different learning rates. Five candidate texts were generated for each source text. The candidate with the highest probability that does not consist of the same letters as the source was chosen for evaluation.

We report the following metrics: BERTScore (Zhang et al., 2019),[15] BLEU (Papineni et al., 2002),[16] ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), and TER (Snover et al., 2006). The scores reported in Table 5 are the mean of 4 results from 4 training runs using the settings we described earlier.

Note that the mT5-base trained on the OST dataset outperformed the other models in both datasets. This, in our opinion, suggests generalizability and high dataset quality. To further assess the impact of our filtering method, we fine-tuned mT5-base on the unfiltered datasets and observed that despite the difference in size, the models fine-tuned on the unfiltered datasets yielded worse performance on the OST dataset and less semantically similar pairs on the TAT dataset. We believe that this is due to the fact that TAT is more carefully constructed using crowdsourcing, and thus the effect of semantic similarity based filtering is less visible. We report the score of mT5-base trained for 3 and 4 epochs on the unfiltered OpenSubtitles2018 (OST-RAW) and Tatoeba (TAT-RAW) respectively. The scores of the model on the test splits started to decrease after those epochs.

We present some generated paraphrase examples in Appendix A, to highlight the success and the failure cases of the fine-tuned models.

[15]https://github.com/Tiiiger/bert_score

[16]https://huggingface.co/spaces/evaluate-metric/bleu

6 Conclusion

We detailed an approach for creating paraphrase datasets from parallel text coprora using machine translation and semantic similarity based filtering. For filtering, we chose a semantic similarity model that kept the most paraphrases in the datasets based on similarity labels we collected from human annotators for a sample of our datasets. We present the paraphrase datasets we created with benchmark results of Text-to-Text Transformer models trained on our datasets across a variety of metrics.

7 Future Work

Our approach results in a high-quality paraphrase dataset, but has a downside of filtering out valid pairs with low lexical similarity depending on the semantic similarity metric used. We plan on combining lexical and semantic similarity into a new filtering metric to obtain a dataset that has more diverse pairs. We will compare the effectiveness of models trained on the current datasets and the diverse dataset in data augmentation for different tasks. Furthermore, we also plan to test the effect of curriculum learning (Bengio et al., 2009) on the newly created diverse datasets, and similar to (Li et al., 2018) we will evaluate the output of the models with the help of human annotators on multiple aspects like clarity, fluency, and semantic similarity.

8 Acknowledgment

This study was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) Grant No: 120E100.

References

Mikko Aulamo, Umut Sulubacak, Sami Virpioja, and Jörg Tiedemann. 2020. OpusTools and parallel corpus diagnostics. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 3782–3789. European Language Resources Association.

OST Test Dataset

Model	Train Dataset	BERTScore Cased	BERTScore Uncased	BLEU	ROUGE-L	METEOR	TER
mT5-base	OST	**89 ± 0.01**	**92.05 ± 0.01**	**46.26 ± 0.09**	**74.8 ± 0.02**	**72.97 ± 0.13**	**36.4 ± 0.04**
trBART	OST	77.8 ± 0.17	87.92 ± 0.13	33.59 ± 0.32	64.65 ± 0.33	62.62 ± 0.45	50.96 ± 0.4
mT5-base	TAT	84.95 ± 0.38	89.23 ± 0.24	29.37 ± 0.83	66.64 ± 0.46	63.14 ± 0.86	49.29 ± 1.13
trBART	TAT	74.21 ± 0.32	85.25 ± 0.28	23.45 ± 0.29	59.32 ± 0.6	54.93 ± 0.5	57.22 ± 0.59

TAT Test Dataset

Model	Train Dataset	BERTScore Cased	BERTScore Uncased	BLEU	ROUGE-L	METEOR	TER
mT5-base	TAT	94.07 ± 0.36	95.75 ± 0.25	61.66 ± 1.34	84.67 ± 0.62	**82.72 ± 0.42**	22.43 ± 1.27
trBART	TAT	84.42 ± 0.33	94.09 ± 0.26	56.58 ± 0.99	81.68 ± 0.54	78.83 ± 0.52	26.69 ± 0.76
mT5-base	OST	**94.47 ± 0.06**	**95.94 ± 0.03**	**63.87 ± 0.44**	**85.18 ± 0.19**	82.46 ± 0.27	**21.41 ± 0.21**
trBART	OST	82.65 ± 0.25	92.47 ± 0.16	48.71 ± 0.69	76.45 ± 0.32	73.26 ± 0.6	34.79 ± 0.28

Table 5: The Performance Scores of Our Models on the Test Datasets. TER score measures distance. The other metrics measure similarity.

OST Test Dataset

Model	Train Dataset	BERTScore Cased	BERTScore Uncased	BLEU	ROUGE-L	METEOR	TER
mT5-base	OST	**89 ± 0.01**	**92.05 ± 0.01**	**46.26 ± 0.09**	**74.8 ± 0.02**	**72.97 ± 0.13**	**36.4 ± 0.04**
mT5-base	OST *(Unfiltered)*	88.89 ± 0.06	91.94 ± 0.04	36.4 ± 0.23	73.87 ± 0.09	72.16 ± 0.16	37.58 ± 0.15
mT5-base	TAT	84.95 ± 0.38	89.23 ± 0.24	29.37 ± 0.83	66.64 ± 0.46	63.14 ± 0.86	49.29 ± 1.13
mT5-base	TAT *(Unfiltered)*	88.95 ± 0.2	92.08 ± 0.14	38.13 ± 0.4	68.39 ± 0.23	65.87 ± 0.22	45.13 ± 0.33

TAT Test Dataset

Model	Train Dataset	BERTScore Cased	BERTScore Uncased	BLEU	ROUGE-L	METEOR	TER
mT5-base	TAT	94.07 ± 0.36	95.75 ± 0.25	61.66 ± 1.34	84.67 ± 0.62	82.72 ± 0.42	22.43 ± 1.27
mT5-base	TAT *(Unfiltered)*	91.61 ± 0.12	93.93 ± 0.09	34.74 ± 0.62	**86.6 ± 0.2**	**84.85 ± 0.23**	**18.23 ± 0.25**
mT5-base	OST	**94.47 ± 0.06**	**95.94 ± 0.03**	**63.87 ± 0.44**	85.18 ± 0.19	82.46 ± 0.27	21.41 ± 0.21
mT5-base	OST *(Unfiltered)*	91.97 ± 0.07	94.2 ± 0.05	37.02 ± 0.16	84.05 ± 0.19	81.59 ± 0.28	22.76 ± 0.32

Table 6: A Comparison Between the Performance of mT5 Model Checkpoints Trained on Our Filtered and Unfiltered Datasets. TER score measures distance. The other metrics measure similarity.

Ahmet Bağcı and Mehmet Fatih Amasyali. 2021. Comparison of turkish paraphrase generation models. In *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–6. IEEE.

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, Ann Arbor, Michigan. Association for Computational Linguistics.

Figen Beken Fikri, Kemal Oflazer, and Berrin Yanikoglu. 2021. Semantic similarity based evaluation for abstractive news summarization. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, pages 24–33, Online. Association for Computational Linguistics.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Emrah Budur, Rıza Özçelik, Tunga Gungor, and Christopher Potts. 2020. Data and Representation for Turkish Natural Language Inference. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8253–8267, Online. Association for Computational Linguistics.

Steven Burrows, Martin Potthast, and Benno Stein. 2013. Paraphrase acquisition via crowdsourcing and machine learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(3):1–21.

Ruisheng Cao, Su Zhu, Chenyu Yang, Chen Liu, Rao Ma, Yanbin Zhao, Lu Chen, and Kai Yu. 2020. Unsupervised dual paraphrasing for two-stage semantic parsing. *arXiv preprint arXiv:2005.13485*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Mathias Creutz. 2018. Open subtitles paraphrase corpus for six languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Seniz Demir, Ilknur Durgar El-Kahlout, and Erdem Unal. 2013. A case study towards turkish paraphrase

alignment. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 188–192.

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.

Sonal Garg, Sumanth Prabhu, Hemant Misra, and G Srinivasaraghavan. 2021. Unsupervised contextual paraphrase generation using lexical control and reinforcement learning. *arXiv preprint arXiv:2103.12777*.

Bahar Karaoğlan, Tarık Kışla, Senem Kumova Metin, Ufuk Hürriyetoğlu, and Katira Soleymanzadeh. 2016. Using multiple metrics in automatically building turkish paraphrase corpus. *Research in Computing Science*, 117:75–83.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. 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 3609–3619.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark. 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.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878, Brussels, Belgium. 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.

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.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia. European Language Resources Association (ELRA).

Louis Martin, Angela Fan, Éric de la Clergerie, Antoine Bordes, and Benoît Sagot. 2020. Muss: multilingual unsupervised sentence simplification by mining paraphrases. *arXiv preprint arXiv:2005.00352*.

Aurélien Max and Guillaume Wisniewski. 2010. Mining naturally-occurring corrections and paraphrases from Wikipedia's revision history. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

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.

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*. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525, Online. Association for Computational Linguistics.

Ali Safaya, Emirhan Kurtuluş, Arda Goktogan, and Deniz Yuret. 2022. Mukayese: Turkish NLP strikes back. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 846–863, Dublin, Ireland. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Yui Suzuki, Tomoyuki Kajiwara, and Mamoru Komachi. 2017. Building a non-trivial paraphrase corpus using multiple machine translation systems. In *Proceedings of ACL 2017, Student Research Workshop*, pages

36–42, Vancouver, Canada. Association for Computational Linguistics.

Brian Thompson and Matt Post. 2020. Automatic machine translation evaluation in many languages via zero-shot paraphrasing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 90–121, Online. Association for Computational Linguistics.

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT — Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Melbourne, Australia. 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, Online. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Shuguang Zhu, Xiang Cheng, Sen Su, and Shuang Lang. 2017. Knowledge-based question answering by jointly generating, copying and paraphrasing. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2439–2442.

A Paraphrase Examples

We present in Table 7, 8 examples of paraphrases generated by the models fine-tuned on our train datasets. An abbreviation of the dataset each model was fine-tuned on is provided in parenthesis. We tried to choose representative examples showing cases of both failure and success.

Source	Ve onu sizden kimse alamaz, beyler. *And no one can take it away from you, gentlemen.*	Woodhouse tatlım biraz daha buza ihtiyacım var. *Woodhouse, honey, I need some more ice.*	Bir sandviçe yetecek kadar malzemem var. *I've got stuff that will be enough for a sandwich.*
mT5-base (OST)	Ve kimse onu sizden alamaz, beyler. *And no one can take it away from you, gentlemen.*	Woodhouse, tatlım, biraz daha buz lazım. *Woodhouse, honey, there is a need for more ice.*	Bir sandviç için yeterli malzemem var. *I've got enough stuff for a sandwich.*
trBART (OST)	beyler ve onu sizden kimse alamaz. *gentlemen and no one can take it away from you.*	woodhouse biraz daha buza ihtiyacım var. *woodhouse, I need some more ice.*	bir sandviçe yetecek kadar malzeme var. *there is enough stuff for a sandwich.*
mT5-base (TAT)	Kimse bunu sizden alamaz, beyler. *No one can take that away from you, gentlemen.*	Woodhouse tatlım biraz daha buza ihtiyacın var. *Woodhouse, honey, you need more ice.*	Sandviçe yetecek kadar malzemem var. *I've got stuff that will be enough for sandwich.*
trBART (TAT)	beyler ve onu sizden kimse alamaz. *gentlemen and no one can take it away from you.*	woodhouse biraz daha buza ihtiyacım var. *woodhouse, honey, there is a need for more ice.*	bir sandviçe yetecek kadar malzeme var. *there is enough stuff for a sandwich.*
mT5-base (OST-RAW)	Kimse onu senden alamaz, çocuklar. *No one can take it away from you, kids.*	Woodhouse, tatlım, biraz daha buz lazım. *Woodhouse, honey, there is a need for more ice.*	Bir sandviç için yeterli malzemem var. *I've got enough stuff for a sandwich.*
mT5-base (TAT-RAW)	Kimse bunu sizden alamaz, beyler. *No one can take that away from you, gentlemen.*	Woodhouse tatlım biraz daha buza ihtiyacın var. *Woodhouse honey you need more ice.*	Sandviçe yetecek kadar malzemem var. *I've got stuff that will be enough for sandwich.*

Table 7: Generated Paraphrases of Examples from the OST Dataset

Source	Tom daha sonra ne yapacağını bilmiyordu. *Tom didn't know what to do next.*	Tom asla tek başına oraya gitmezdi. *Tom would never go there by himself*	İlk olarak ne yapacaklarını merak ettiler. *They wondered what they would do first.*
mT5-base (OST)	Tom ne yapacağını bilmiyordu. *Tom didn't know what to do.*	Tom oraya tek başına gitmezdi. *Tom wouldn't go there by himself.*	Önce ne yapacaklarını merak ettiler. *They wondered what they would do before.*
trBART (OST)	tom bundan sonra ne yapacağını bilmiyordu. *tom didn't know what to do next.*	tom oraya hiç gitmezdi. *tom never went there.*	ilk olarak ne yapacaklarını merak ediyorlar. *they are wondering what they're going to do first.*
mT5-base (TAT)	Tom sonra ne yapacağını bilmiyordu. *Tom didn't know what to do next.*	Tom oraya asla tek başına gitmez. *Tom never goes there by himself.*	İlk başta ne yapacaklarını merak ettiler. *They wondered what they were going to do at first.*
trBART (TAT)	tom bundan sonra ne yapacağını bilmiyordu. *tom didn't know what to do next.*	tom oraya hiç gitmezdi. *tom never went there.*	ilk olarak ne yapacaklarını merak ediyorlar. *they are wondering what they're going to do first.*
mT5-base (OST-RAW)	Tom bundan sonra ne yapacağını bilmiyordu. *tom didn't know what to do next.*	Tom oraya hiç tek başına gitmedi. *Tom didn't go there by himself.*	Önce ne yapacaklarını merak ediyorlar. *They are wondering what they would do before.*
mT5-base (TAT-RAW)	Tom bundan sonra ne yapacağını bilmiyordu. *Tom didn't know what to do next.*	Tom oraya asla tek başına gitmez. *Tom never goes there by himself.*	Önce ne yapacaklarını merak ettiler. *They wondered what they would do before.*

Table 8: Generated Paraphrases of Examples from the TAT Dataset

Second-order Document Similarity Metrics for Transformers

Jarkko Lagus
Department of Computer Science
University of Helsinki
jarkko.lagus@helsinki.fi

Niki Loppi
NVIDIA
nloppi@nvidia.com

Arto Klami
Department of Computer Science
University of Helsinki
arto.klami@helsinki.fi

Abstract

The similarity of documents represented using static word embeddings is best measured using second-order metrics accounting for the covariance of the embeddings. Transformers provide superior representations for words compared to static embeddings, but document representation and similarity evaluation are currently often done using simple mean pooling. We explain how the second-order metrics can be used also with transformers, and evaluate the value of improved metrics in this context.

1 Introduction

Many NLP models rely on pretrained representations, either static embeddings (e.g. Word2Vec by Mikolov et al. (2013)) or context-aware models like transformers (e.g. BERT by Devlin et al. (2018)) that process text sequentially but still represent each word or subword with a fixed-dimensional latent representation. These models are then fine-tuned to solve specific tasks, by continuing to train the representations while optimizing for the task performance.

Longer documents cannot be directly modeled by most transformers, but a representation can be obtained by processing them in smaller units (e.g. sentences) and then *pooling* the representations of the individual words. The pooling method is often a simple mean or max pooling, but despite the simplicity, such approaches work relatively well both with static embeddings (Wieting et al., 2015; Arora et al., 2017; Gupta et al., 2020) and transformers (Devlin et al., 2018; Reimers and Gurevych, 2019).

For instance, Torki (2018); Nikolentzos et al. (2017); Muzellec and Cuturi (2018) and Lagus et al. (2019) have shown that for static embeddings the similarities can be more accurately captured by accounting for the covariance structure of the word matrix using so-called second-order metrics. We study whether this holds also for transformers. This is not obvious upfront, since transformers process documents sequentially and hence capture some document-level information already in (sub)word representations, and directly fine-tuning the representations for accurate document comparison may enable even mean pooling to capture some of the same information the second-order representations use.

We explain how model-agnostic second-order metrics can be used with transformers, provide details for fine-tuning both full-rank (Torki, 2018) and low-rank (Mu et al., 2017; Yang et al., 2018; Lagus et al., 2019) metrics using the new pooling functions, and evaluate them in three tasks. We focus on precomputable document representations, not relying on similarity comparisons that require cross-encoding the document pairs like in the original BERT paper by Devlin et al. (2018). This makes the methods suitable for online processing and enables caching representations in a database. Second-order metrics have computational overhead in isolation, but the fine-tuning process is dominated by other parts of the model and hence the added computation time is small, even for low-rank metrics that require propagating gradients through singular-value decomposition (SVD). We show that second-order metrics improve document similarity comparison in two languages and for two transformer models, but do not help for sentence similarity.

2 Document Metrics

We denote by $A \in \mathbb{R}^{n \times d}$ a matrix that collects the d-dimensional representations of the n words occurring in the document as its rows.

The first-order metrics compress A into a d-dimensional vector, typically using the mean $a = \frac{1}{n}\mathbf{1}^T A$, where $\mathbf{1}$ is n-dimensional vector of ones, and compare the documents e.g. by cosine similarity $S_{cos}(a, b) = (a \cdot b)(\|a\|\|b\|)^{-1}$ where b denotes the mean vector for another document B (typically with different n). See Arora et al. (2017) for details

(e.g. weighting) for static embeddings and Reimers and Gurevych (2019) for use with transformers.

The second-order metrics are based on covariance-like products $A^TA \in \mathbb{R}^{d\times d}$ capturing both variance and correlation between the representation dimensions. Several slightly different metrics have been proposed: Torki (2018) vectorized the covariance and combined it with the mean pooling, Lagus et al. (2019) derived a second-order metric from pair-wise cosine similarity of all word pairs in the two documents, and Muzellec and Cuturi (2018) used Wasserstein metric to compare second-order representations. Next, we describe the specific metric of Lagus et al. (2019) that supports also low-rank computation proposed for computational reasons but note that the other metrics can be implemented as minor modifications.

The second-order metrics compare A^TA and B^TB. A metric normalized to range $[-1, 1]$ can be conveniently expressed as $S_F(A^TA, B^TB)$ using the general similarity measure

$$S_F(X, Y) = \frac{\langle X, Y\rangle_{\mathrm{F}}}{\|X\|_{\mathrm{F}}\|Y\|_{\mathrm{F}}}, \qquad (1)$$

where $\langle X, Y\rangle_{\mathrm{F}} = \mathrm{Tr}\left(X^TY\right)$ is the Frobenius inner product and $\|X\|_{\mathrm{F}} = \sqrt{\langle X, X\rangle_{\mathrm{F}}}$.

The obvious drawback of the metric is that A^TA has $O(d^2)$ elements, compared to $O(d)$ of the mean representation. Lagus et al. (2019) showed that forming the covariance matrix can be avoided by computing SVD of A: If $A = U\Sigma V^T$ then $A^TA = V\Sigma U^TU\Sigma V^T = V\Sigma^2V^T$, and the trace term can be evaluated using the SVDs of the two document matrices. They also demonstrated empirically that a low-rank approximation for the metric, using $A_k = \Sigma_kV_k^T \approx A$ with $k \ll \min(n, d)$ components has a regularizing effect while reducing computation time compared to full-rank matrices.

3 Fine-tuning Second-order Metrics

Fine-tuning of transformers is done end-to-end for a model that combines four parts: the embedding model $E(\cdot)$, a pooling function $P(\cdot)$, the similarity metric $S(\cdot, \cdot)$ and some eventual loss function $L(\cdot, \cdot)$ that compares the similarities with ground truths. If we denote by $A = E(d_A)$ the matrix formed by the embedding model processing a text sequence and by $\tilde{S}$ the ground truth similarity, the objective to be trained with respect to parameters of $E(\cdot)$, starting with the pretrained values, is

$$L(S(P(E(d_A)), P(E(d_B))), \tilde{S}(d_A, d_B)).$$

3.1 Pooling for Second-order Metrics

The current modeling pipelines building on mean pooling $P_{mean}(A) = \frac{1}{n}\mathbf{1}^TA$ can be re-used with second-order metrics, by re-writing the metrics as a combination of a generalized pooling function and a distance measure. For this, we assume $P(A)$ to be any function that transforms A to a fixed-dimensional representation (vector or matrix). Then we can define the new pooling functions

$$P_{cov}(A) = A^TA \quad \text{and} \quad P_{svd}(A, k) = \sqrt{\Sigma_k}V_k^T,$$

where Σ_k and V_k refer to matrices that retain the singular values and vectors corresponding to k largest values. In the experiments, we also use additional method derived from $P_{cov}(\cdot)$ named, $P_{cov}(\cdot, k)$ where the dimensionality reduction to k dimensions is done after the fine-tuning in the prediction phase. Then the three metrics for document comparison can be written as

$$\begin{aligned}\text{mean} &: S_{cos}(P_{mean}(A), P_{mean}(B)),\\ \text{full-rank} &: S_F(P_{cov}(A), P_{cov}(B)),\\ \text{low-rank} &: S_F(P_{svd}(A, k), P_{svd}(B, k)).\end{aligned}$$

This unified formulation makes it easy to implement the second-order metrics as part of standard processing pipelines. The implementation of the second-order metrics, compatible with the *sentence-transformers* library by Reimers and Gurevych (2019), is made available on GitHub[1].

The size of $P_{cov}(\cdot)$ is quadratic in d and hence can be large, which is generally considered a challenge in the case of static embeddings. In the context of transformers, however, this is insignificant since we are anyway fine-tuning a typically very large model. The practical computation is hence largely dominated by the other parts of the pipeline (see Section 4 for empirical validation).

3.2 Differentiable SVD

Fine-tuning for full-rank second-order metrics does not require special treatment, but for low-rank metrics we need to propagate gradients through the singular value decomposition used for approximating A^TA as described in Section 2. Deep learning frameworks offer differentiable SVD implementations out-of-the-box, providing multiple approximate algorithms with varying properties. While the forward pass through SVD is relatively fast and stable for all choices, the backward pass is

[1] https://github.com/jalagus/second-order-transformers

more costly and prone to instabilities arising from ill-conditioned document matrices. We found the CPU version of `torch.svd_lowrank`, implementing the algorithm of Halko et al. (2011), to be the most stable and hence use that.

Another challenge is the sign ambiguity over the matrices U and V. The pooling function $P_{svd}(\cdot, k) = \sqrt{\Sigma_k} V_k^T$ is hence not unique which causes issues with the distance computation; some components might point to opposite directions while still encoding the same information.

To address these issues we explicitly reconstruct the covariance matrix $A^T A$ to improve backward pass stability and remove the sign ambiguity. The pooling is thus implemented as $P_{svd}(A, k) = U_k \Sigma_k V_k^T$. This has no practical effect on computation speed as it is dominated by other components of the full model. Since $A^T A$ is symmetric, the U and V matrices will be identical and it is enough to save the matrix $D = \sqrt{\Sigma_k} V_k^T$ and reconstruct the full covariance inference-time using $D^T D$. The space requirement is then $O(kd)$.

The same space reduction for the representation can be obtained by fine-tuning using the full-rank covariance and computing the low-rank representation using SVD only after the training is done. This approach does not need a differentiable SVD, but as we will later show it performs worse in practice.

4 Experiments

We demonstrate second-order metrics in three example cases; sentence similarity on the STS benchmark (Cer et al., 2017) and two document similarity tasks. For all experiments, we compare the three alternative pooling methods and provide results for alternative embedding methods.

4.1 STS Benchmark: Sentence Similarity

Even though our main goal is to provide tools for longer documents, we also evaluate the methods in the STS sentence matching benchmark. The STS experiment was conducted using the *sentence-tranformers* library, by replacing only the pooling method and the distance computation according to the proposed metric. We compared three pooling operations (mean, covariance, and SVD with $k = 1$) for three transformers, BERT (Devlin et al., 2018), TinyBERT (Turc et al., 2019), and RoBERTa (Liu et al., 2019), and for completeness include also results on static GloVe embeddings (Pennington et al., 2014). The *sentence-tranformers* library handles the input tokenization and the default mean pooling serves as a baseline representing the current practice in the field.

Table 1: Sentence similarity on STS benchmark.

Model	Pooling	IC	MC	WT
BERT	$P_{cov}(\cdot)$	0.5957	0.8831	16.59
BERT	$P_{svd}(\cdot, 1)$	0.5900	0.8838	336.37
BERT	$P_{mean}(\cdot)$	0.5932	0.8775	11.67
TinyBERT	$P_{cov}(\cdot)$	0.6932	0.7975	3.29
TinyBERT	$P_{svd}(\cdot, 1)$	0.6930	0.7962	20.38
TinyBERT	$P_{mean}(\cdot)$	0.6937	0.7823	3.20
RoBERTa	$P_{cov}(\cdot)$	0.6463	0.8874	16.32
RoBERTa	$P_{svd}(\cdot, 1)$	0.6446	0.8875	312.63
RoBERTa	$P_{mean}(\cdot)$	0.6500	0.8857	12.03
GloVe	$P_{cov}(\cdot)$	0.7425	0.7425	-
GloVe	$P_{svd}(\cdot, 1)$	0.6517	0.6517	-
GloVe	$P_{mean}(\cdot)$	0.7163	0.7163	-

Table 1 shows the Pearson correlation with true similarity. Here IC denotes the initial correlation before fine-tuning and MC denotes the maximum correlation during fine-tuning. We report results on the development set, following the practice of cross-encoding with `[SEP]` tags by (Devlin et al., 2018) and using the *sentence-transformers* library for easy reproduction. We ran each configuration for 15 epochs using a batch size of 16. WT denotes the wall-time (minutes) taken to complete the entire fine-tuning process. All reported numbers are averages of five complete fine-tuning runs on different seeds, and our baseline results with $P_{mean}(\cdot)$ are in line with those reported in *sentence-transformers*.

For all embedding methods, using a second-order metric improves the similarity compared to mean pooling, but the gain is considerably smaller for the three transformers (below 1%) compared to the static embeddings (3.6%). For transformers the low-rank metrics perform identically with full-rank, whereas for GloVe using full-rank is preferred, matching the result of Lagus et al. (2019) for STS. Without fine-tuning the transformers are worse than static embeddings, highlighting the well known importance of fine-tuning them. In conclusion, second-order metrics can be used with transformers already for sentence comparison, but the gain is very small and due to the need of computing SVD on CPU the computation is slower.

4.2 Full-length document experiments

Second-order metrics are expected to be more useful for longer documents, and hence we evaluate them on two document similarity tasks. In both experiments, triplet loss (Schultz and Joachims,

Table 2: Document similarity: Finnish news

Model	Pooling	Initial Acc	Acc
FinBERT	$P_{mean}(\cdot)$	0.499	0.626
FinBERT	$P_{cov}(\cdot)$	0.325	0.663
FinBERT	$P_{cov}(\cdot, 1)$	-	0.484
FinBERT	$P_{cov}(\cdot, 5)$	-	0.591
FinBERT	$P_{svd}(\cdot, 1)$	-	0.644
FinBERT	$P_{svd}(\cdot, 5)$	-	0.644
fastText	$P_{mean}(\cdot)$	-	0.194
fastText	$P_{cov}(\cdot)$	-	0.073

Table 3: Document similarity: Patent retrieval

Model	Pooling	Initial Acc	Acc
TinyBERT	$P_{mean}(\cdot)$	0.553	0.885
TinyBERT	$P_{cov}(\cdot)$	0.606	0.901
TinyBERT	$P_{cov}(\cdot, 1)$	-	0.836
TinyBERT	$P_{cov}(\cdot, 5)$	-	0.867
TinyBERT	$P_{svd}(\cdot, 1)$	-	0.844
TinyBERT	$P_{svd}(\cdot, 5)$	-	0.879
fastText	$P_{mean}(\cdot)$	-	0.566
fastText	$P_{cov}(\cdot)$	-	0.592

2004) is used as the optimization target and hyperparameter optimization is done using grid search, performing computation on CPU because of instability of SVD computations on GPU.

Finnish news data We use data from the Finnish national broadcasting company *Yle* [2], of news articles written in easy-to-read Finnish, a morphologically rich language. We form an artificial task where each article is split into two equal-sized parts and the goal is to retrieve the correct second part (amongst the set of all second parts) using the first part as a query. For the triplet loss, we use the second half of a random document from the training set as the negative anchor.

For fine-tuning, we split the dataset of 600 triplets into 500 training samples and 100 validation samples. We use FinBERT (Virtanen et al., 2019) model and fine-tune it for 15 epochs, evaluating the final accuracy on a fresh set of 1500 samples. For baseline computations, we use Finnish fastText embeddings.

Patent data As a real document similarity comparison task, we consider a patent retrieval task. When patents are applied, multiple kinds of prior art might lead to rejection. Here we consider two types of prior art, namely X and A citations. The X citations are prior work that can alone lead to a rejection, while the A citations describe the state of the art, but are not immediate reasons for rejection. An ideal model would hence rank the X citations ahead of A citations.

Patent documents are split into two main parts, claims and description. We only use the claims part – containing approximately 2,100 character per document – that defines the exact claims of the invention, leaving out the more free-form description. For training, we split a proprietary dataset acquired from the United States Patent and Trademark Office of 1000 triplets into 800 training samples and 200 validation samples. In each triplet, the anchor is the patent document, the positive sample is any X citation of that patent and the negative sample is any A citation of that same patent. We fine-tune TinyBERT model for 15 epochs and evaluate it on a fresh set of 1000 documents. Static embeddings are used as a baseline.

Results The results for both tasks, reported in Tables 2 and 3, are similar. The main observations are (a) transformers clearly outperform static embeddings, (b) fine-tuning for document similarity comparisons improves accuracy dramatically, and (c) the second-order metrics outperform the standard mean pooling clearly, with improvements of 3.7 and 1.6 percentage points for the two tasks ($P_{cov}(\cdot)$ vs $P_{mean}(\cdot)$). In conclusion, the second-order metrics are useful also in the case of transformers.

On both tasks, the total computation time for the low-rank models is only approximately 5% higher (not shown), due to need for fewer iterations for SVD, and hence the choice of the metric can be made purely based on the accuracy. Here the full-rank ($P_{cov}(\cdot)$) metrics slightly outperformed the low-rank ones ($P_{svd}(\cdot, k)$), but the latter may still be beneficial due to smaller representations. Finally, we see that extracting a low-rank representation from a model fine-tuned for the full-rank metric (denoted by $P_{cov}(\cdot, k)$) is worse than directly fine-tuning for the low-rank metric. As there is no notable difference in computational cost, we do not recommend doing this.

5 Conclusions

Transformers provide richer representations for text compared to static embeddings. For documents, the current practice is to use averages of the word

[2] http://urn.fi/urn:nbn:fi:lb-2019121205

representations for similarity comparisons, which is naive compared to the richer representations and distance metrics used for static embeddings.

We investigated the use of second-order metrics in the case of transformers, showed how they can be implemented into existing pipelines using pooling functions, and empirically demonstrated consistent improvement in similarity comparisons. The gain is smaller than with static embeddings but especially for longer documents still clear and consistent across different setups. Even though the improvement is not particularly large, the metrics are easy to use and hence we recommend practitioners to evaluate performance for both first-order and second-order metrics – at least the full-rank one that does not have computational overhead – and select the best metric on a validation data.

Acknowledgments

This work was supported by the Academy of Finland Flagship programme: Finnish Center for Artificial Intelligence FCAI. NL contributed under the NVIDIA AI Technology Center (NVAITC) Finland programme. In addition, we thank IPRally Technologies Oy for providing the data for the experiments done on patent data.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of International Conference on Learning Representations*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

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

Vivek Gupta, Ankit Saw, Pegah Nokhiz, Praneeth Netrapalli, Piyush Rai, and Partha Talukdar. 2020. P-SIF: Document embeddings using partition averaging. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7863–7870.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Jarkko Lagus, Janne Sinkkonen, and Arto Klami. 2019. Low-rank approximations of second-order document representations. In *Proceedings of the 23rd Conference on Computational Natural Language Learning*.

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*.

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

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. Representing sentences as low-rank subspaces. *arXiv preprint arXiv:1704.05358*.

Boris Muzellec and Marco Cuturi. 2018. Generalizing point embeddings using the wasserstein space of elliptical distributions. In *Advances in Neural Information Processing Systems*, pages 10258–10269.

Giannis Nikolentzos, Polykarpos Meladianos, François Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. 2017. Multivariate Gaussian document representation from word embeddings for text categorization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 450–455.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

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*. Association for Computational Linguistics.

Matthew Schultz and Thorsten Joachims. 2004. Learning a distance metric from relative comparisons. In *Advances in neural information processing systems*, pages 41–48.

Marwan Torki. 2018. A Document Descriptor using Covariance of Word Vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 527–532.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is

not enough: BERT for Finnish. *arXiv preprint arXiv:1912.07076*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Ziyi Yang, Chenguang Zhu, and Weizhu Chen. 2018. Parameter-free sentence embedding via orthogonal basis. *arXiv preprint arXiv:1810.00438*.

Semantic Similarity-Based Clustering of Findings From Security Testing Tools

Phillip Schneider[1*], **Markus Voggenreiter**[2*], **Abdullah Gulraiz**[1*] **and Florian Matthes**[1]
[1]Technical University of Munich, Department of Computer Science, Germany
[2]Siemens Technology & LMU Munich, Germany
{phillip.schneider, abdullah.gulraiz, matthes}@tum.de
markus.voggenreiter@siemens.com

Abstract

Over the last years, software development in domains with high security demands transitioned from traditional methodologies to uniting modern approaches from software development and operations (DevOps). Key principles of DevOps gained more importance and are now applied to security aspects of software development, resulting in the automation of security-enhancing activities. In particular, it is common practice to use automated security testing tools that generate reports after inspecting a software artifact from multiple perspectives. However, this raises the challenge of generating duplicate security findings. To identify these duplicate findings manually, a security expert has to invest resources like time, effort, and knowledge. A partial automation of this process could reduce the analysis effort, encourage DevOps principles, and diminish the chance of human error. In this study, we investigated the potential of applying Natural Language Processing for clustering semantically similar security findings to support the identification of problem-specific duplicate findings. Towards this goal, we developed a web application for annotating and assessing security testing tool reports and published a human-annotated corpus of clustered security findings. In addition, we performed a comparison of different semantic similarity techniques for automatically grouping security findings. Finally, we assess the resulting clusters using both quantitative and qualitative evaluation methods.

1 Introduction

The automation of security tests is a common practice for software engineering projects that apply software development and operations (DevOps) practices. Different security tools employ different perspectives to scan a software artifact as part of Continuous Integration or Continuous Deployment (CI/CD) pipelines, producing semi-structured reports of security findings. While this approach fosters DevOps principles, reduces manual effort, and shifts security efforts to the earlier stages of development, it also comes at a cost.

Since security testing tools often have an overlapping scanning coverage, duplicates or nearly identical findings are unavoidable. Further, considering that each iteration brings new security findings, identifying duplicate security findings is essential to achieve a reliable overview. In this context, it is important to note that we define *duplicates* as findings that point out the exact same security problem, potentially occurring at multiple locations in the software. Exemplary for that would be an SQL injection vulnerability at multiple locations of a web interface. Amongst multiple other activities, the identification of duplicates is traditionally addressed by a team member with security domain knowledge, a so-called security professional, before looping back the security findings to development to improve the software security-wise (Simpson, 2014). Taking the frequency of new reports and the number of findings throughout all security tests into account, an entirely manual analysis is unfeasible, prone to human error, and violates fundamental DevOps principles.

Natural Language Processing (NLP) has been shown to be effective in analyzing and clustering textual data from various application domains, such as medicine, linguistics, and software engineering (Demner-Fushman and Lin, 2006; Majewska et al., 2018; Aggarwal et al., 2017). Although security tool reports contain highly domain-specific text, it seems promising to investigate NLP techniques for automatically grouping findings into problem-oriented clusters, which can assist security professionals in their analyses. To our best knowledge, no studies specifically focus on the machine-generated finding texts produced by security scanning tools.

* The first three authors have contributed equally.

Addressing this research gap, we evaluated the performance of three common semantic similarity techniques. The selected techniques originate from knowledge-based, corpus-based, and neural network-based methods. Our main contributions are twofold:

1. We publish a human-annotated corpus of clustered security findings along with the annotation tool used by the security professionals.

2. We perform an in-depth analysis of three popular semantic similarity techniques for clustering security findings, followed by a quantitative and qualitative evaluation of the results.

The remainder of this paper is structured as follows. Section 2 presents background information on security scanning tools and gives an overview of related work on applying NLP techniques in the software engineering domain. Section 3 describes the employed two-stage research approach for the dataset construction and experimental evaluation. We report the clustering results, discuss our observations, and outline the limitations in Section 4, Section 5, and Section 6, respectively. Section 7 concludes the paper with a summary and an outlook toward future work.

2 Background and Related Work

This section provides background information on security testing tools and security finding reports in DevOps. Furthermore, we mention related studies concerning the application of NLP techniques in the software engineering domain.

To tackle the challenge of duplicates in security reports, we first establish the definition of what duplicate security findings are. We consider two findings to be duplicates if they describe the exact same problem at any location of the software. Consequently, the same issue, e.g., an SQL injection, could occur at multiple places but would be considered a duplicate. Besides the problem-based approach, other strategies for describing duplicates can also incorporate the location of a finding or its underlying solution. The selection of a strategy in this area highly depends on the subsequent actions on the dataset.

Furthermore, it is necessary to explain the activities that generate security reports that contain duplicate findings. Security testing can be categorized according to multiple properties depending on the testing strategy, involved testers, tested components, and numerous others. We limit our categorization to those security tests that can be automated in pipelines and scan an actual part of the product. Further, we categorize them into two major categories: tests that examine the static elements of the software (e.g., code, configuration, or dependencies) are called static application security testing (SAST) and tests performed against the dynamic, actually running application are called dynamic application security testing (DAST). This separation represents a clear distinction, as static testing can only guess whether a finding is actually affecting the software, while dynamic techniques directly identify the exploitable security finding.

From our analysis of the literature on security findings management, we found that there are no NLP-related publications that focus on the identification of duplicate security findings. However, a number of NLP methods have been successfully applied to related subdomains in the software engineering field. For example, Kuhn et al. (2007) use latent semantic indexing (*LSI*) and clustering to analyze linguistic information found in source code, such as identifier names or comments, to reveal topics and support program comprehension. In a study from Schneider (2020), a corpus of app reviews with comments about a variety of software issues is clustered into topics with problem-specific issue categories. Another study from Eyal Salman et al. (2018) focuses on automatically forming semantic clusters of functional requirements based on cosine similarity with a corpus of documents containing software requirements specifications. The authors conduct an empirical evaluation of agglomerative hierarchical clustering using four open-access software projects. In order to assess the software quality of programs, Tan et al. (2011) apply a hierarchical cluster algorithm to create problem-oriented clusters, reducing the effort needed to review the code. The study shows that semantic clusters are an effective technique for defect prediction.

3 Method

In order to achieve our objective of investigating semantic similarity techniques for clustering findings from security testing reports, we constructed a human-annotated dataset. This annotated corpus consists of 1351 SAST and 36 DAST findings. The two-stage process with dataset construction as well as experimental evaluation is explained in the following subsections.

3.1 Dataset Construction

To quantify the performance of different semantic similarity techniques, a ground-truth benchmark dataset is required, enabling the comparison between human-labeled clusters and the predictions of the semantic similarity algorithms. Therefore, we asked two security professionals from the industry to annotate semantically duplicate findings in a given list of security reports. Due to the significant differences in perspective between SAST and DAST reports, we decided to construct two separate datasets, each of which comprising reports from only one testing type.

A major challenge in constructing such a dataset is the content of the security tool reports. Security tool reports are often exported as JSON files containing security finding objects. Across different tools, these reports utilize different schemas, resulting in different property names referring to the same finding feature (e.g., *description*, *FullDescription*, *text*, *Message*, or *details*). For the construction, the security professionals consolidate semantically duplicate findings from all tool reports of a testing iteration based on certain features, e.g., description, location, or unique identifier. Therefore, they need to find the feature in the respective tool schema and compare it to the other findings. Manually annotating such a dataset would require them to memorize $N \times M$ property names when identifying N features across M distinct security testing reports. To enhance efficiency and reduce manual, repetitive work, we developed the Security Findings Labeler (*SeFiLa*).[1] This tool allows security professionals to upload reports from different security tools and conveniently group all findings into named clusters.

The initial, unconsolidated reports of the dataset were generated by scanning the open-source, vulnerable web application JuiceShop[2] with seven SAST tools and two DAST tools. For reproducibility reasons, we solely selected tools free of charge that can be reasonably automated in real-world software development pipelines. We selected Anchore, Dependency Check, Trivy, HorusSec, Semgrep, CodeQL, and Gitleaks as SAST tools. For DAST, we selected Arachni and OWASP ZAP. Fundamental information about each tool can be found in Table 5 in the appendix. From each tool, one testing report was taken for the dataset. The security professionals assigned findings to named clusters representing the same security problem. This process was aided by features like the CVE-ID (common vulnerabilities and exposures) which provides an identifier and a reference-method for publicly known security vulnerabilities. Other helpful features are descriptions and solutions generated by the testing tools. After all findings were assigned to clusters, the dataset comprising our baseline for duplicate identification was completed. The dataset and the code to run the test cases were published in a public GitHub repository.[3]

3.2 Evaluation Procedure

For conducting the evaluation, we investigated semantic similarity methods proposed in the literature and chose three popular techniques that are often used as baseline models: knowledge graph-based similarity with *WordNet* (Miller, 1995), *LSI* (Landauer and Dumais, 1997), and *SBERT* (Reimers and Gurevych, 2019). To evaluate the semantic similarity techniques, we extracted all findings from the security testing tool reports and concatenated selected features from them to form problem-specific finding strings. We applied the three chosen semantic similarity techniques to the finding strings to determine those that are semantically similar. Since semantic similarity between two finding strings is calculated as a score between 0 and 1 where 1 indicates highest similarity, we established a *similarity threshold* for each experiment. This threshold defines the value above which two finding strings are deemed to be semantically similar. Findings corresponding to these similar finding strings are then grouped to form predicted clusters. Implementation-wise, predicted and ground-truth clusters both consist of unique integer sequences, each integer representing a finding from the dataset.

Before the clusters were compared with each other in the quantitative evaluation, we encountered the need for *transitive clustering* of findings. In certain cases, the problem description of two findings was identical, but it was repeated in one finding for multiple instances, leading to a discrepancy in text length. Since the similarity depends on the similarity of the finding strings, we encounter the following example predictions with *Similar Findings* listed in descending order of semantic similarity scores with the corresponding *Finding* identifier:

[1] https://github.com/abdullahgulraiz/SeFiLa

[2] https://owasp.org/www-project-juice-shop/

[3] https://github.com/abdullahgulraiz/SeFiDeF

$$\{Finding: 1,\ Similar\ Findings: \{1, 2, 4\}\}$$

$$\{Finding: 2,\ Similar\ Findings: \{2, 1, 3, 5\}\}$$

Let us assume that findings $\{1, 2, 3\}$ contain the same problem description, although it appears once in $Finding$ 1, two times in $Finding$ 2, and three times in $Finding$ 3. While $Finding$ 1 is found similar to findings $\{1, 2, 4\}$, its similarity score with respect to $Finding$ 3 is below the clustering threshold due to the different text length. However, $Finding$ 2 does have $Finding$ 3 in its set of similar findings. If $Finding$ 3 is similar to $Finding$ 2, it should also be similar to $Finding$ 1, regardless of repetitive text. Therefore, even though $Finding$ 3 exists only in the set of similar findings for $Finding$ 2, it should appear in the final set of similar findings of $Finding$ 1 as well. In our initial clustering experiments and discussions with the security professional, we observed that while lowering the similarity threshold led to many false positive predictions, transitive clustering improved the results without changing the similarity threshold. Therefore, we apply the transitive property to consider findings as semantically related through *intermediate* findings. This causes the above predictions to become:

$$\{Finding: 1, SimilarFindings: \{1, 2, 3, 4, 5\}\}$$

$$\{Finding: 2, SimilarFindings: \{1, 2, 3, 4, 5\}\}$$

After transitive clustering, we removed the duplicate clusters from predictions and evaluated the final predictions against the ground-truth clusters.

Table 1 shows a contingency matrix that illustrates possible outcomes when comparing clusters from predictions (P) with clusters from the ground-truth dataset (Q). The number of occurrences of these outcomes is used to calculate the metrics of *precision*, *recall*, and *F-score*.

	Predictions (P)	
	Clusters in P	Clusters not in P
Ground-truth (Q)		
Clusters in Q	True Positive (TP)	False Negative (FN)
Clusters not in Q	False Positive (FP)	True Negative (TN)

Table 1: Contingency matrix of predicted clusters P and ground-truth clusters Q.

The *precision* (Hossin and Sulaiman, 2015) measures positive patterns correctly predicted from the total predicted patterns in a positive class. In our experiments, it measures the ratio of correct cluster predictions to all predictions. Higher precision indicates that less false positive predictions appeared in the results. It is calculated as:

$$Precision = \frac{TP}{TP + FP}$$

The *recall* (Hossin and Sulaiman, 2015) is used to measure the fraction of correctly classified positive patterns. In our experiments, it represents the ratio of correctly predicted clusters to all ground-truth clusters. A high recall value thus indicates that the semantic clustering results retrieve many ground-truth clusters of the security professional.

$$Recall = \frac{TP}{TP + FN}$$

The *F-score*, also known as Dice Measure (Dice, 1945), calculates the harmonic mean between precision and recall. It balances both metrics to provide an overall performance overview.

$$F - score = \frac{2 * TP}{2 * TP + FP + FN}$$

In addition to the quantitative evaluation with performance measures, we collected qualitative feedback from the security professionals on incorrectly clustered findings. We limited the information about each finding to the finding strings used as input for the NLP techniques and asked for possible reasons for the incorrect clustering. This created a list of reasons that led to poor duplicate identification from the perspective of a domain-aware security professional. Finally, each incorrect cluster is associated with at least one reason for the incorrect clustering, providing insights into the different challenges and their prevalence in the results. The evaluation was aided by *SeFiLa* for annotation of the findings, assignment of reasons, and documentation.

4 Experiments

4.1 Dataset Description

After labeling the exported security findings with our annotation tool *SeFiLa*, the security professionals provided us with two datasets, namely the manually grouped SAST and DAST findings. The descriptive statistics of both datasets are summarized in Table 2. We observe that SAST findings

Statistic	SAST	DAST
Number of clusters	183	10
Number of findings	1351	36
Avg. findings per cluster	7	3
Avg. characters per finding	302	471
Min. findings per cluster	1	1
Max. findings per cluster	408	25

Table 2: Data records from static analysis security tools (SAST) and dynamic analysis security tools (DAST).

are far more frequent, making up 97.4% of the total findings. The number of formed clusters for the SAST findings is significantly higher than for DAST findings. While both datasets had clusters with only one finding, the maximum cluster size was by far larger in the SAST dataset. Despite these discrepancies, the average number of findings per cluster is not too different between the datasets, ranging from a mean value of 3 for DAST to a mean value of 7 for SAST findings. In addition, DAST finding texts are more verbose since they contain 169 more characters on average. To investigate the potential of semantic similarity techniques, constructing the finding string from the finding features is crucial. Analyzing the initial dataset, we identified that solely a single feature describing the finding is consistently found across all SAST tools. For the DAST findings, multiple features, including the description, a name, and even a solution/mitigation, were consistently found across all findings. Furthermore, we observed that DAST features are sufficiently verbose to comprehend the problem from their finding string and thereby contain enough semantic content for NLP. Contrarily, we find SAST features to be very brief, for that matter, making it almost impossible to understand a finding just from the finding string.

To counteract the limitation of very short SAST finding strings, we make use of CVE-IDs to increase the textual content of SAST finding strings. By leveraging the CVE identifier present in some findings, we concatenated finding strings of various machine-generated descriptions with the same CVE-ID. This allows for more semantic content and longer descriptions about the underlying problem. We used the concatenated finding strings as input to the NLP-based similarity techniques.

This step led us to construct a total of four corpora with finding strings from both SAST and DAST datasets for the identification of duplicate findings, as listed below:

- ***SAST-D*:** consists only of SAST finding descriptions
- ***SAST-ConcD*:** consists of concatenated SAST finding descriptions with the same CVE-ID
- ***DAST-NDS*:** consists of concatenated DAST finding names, descriptions, and solution texts
- ***DAST-D*:** consists only of DAST finding descriptions

4.2 Evaluation Results

The summary of the quantitative results achieved when applying semantic clustering using a technique from each category of semantic similarity methods to each of the four corpora is presented in Table 3. The experiments were performed for similarity thresholds $0.1 \leq$ and ≤ 0.95. The performance metric values for the experiment with the highest F-score are reported.

Technique	Corpus	Metrics		
		F-score	Precision	Recall
SBERT	*SAST-D*	0.709	0.621	0.825
	SAST-ConcD	0.797	0.701	0.923
	DAST-NDS	**0.857**	**0.818**	**0.900**
	DAST-D	0.857	0.818	0.900
LSI	*SAST-D*	0.739	0.658	0.842
	SAST-ConcD	**0.816**	**0.734**	**0.918**
	DAST-NDS	0.857	0.818	0.900
	DAST-D	0.857	0.818	0.900
KG	*SAST-D*	0.659	0.556	0.809
	SAST-ConcD	0.777	0.676	0.913
	DAST-NDS	0.727	0.667	0.800
	DAST-D	0.727	0.667	0.800

Table 3: Summary table of performance metrics (highlighted results show the best performing techniques for SAST and DAST).

4.2.1 Comparison of Semantic Similarity Techniques

Figure 1 and Figure 2 show the F-scores of different technique-corpus combinations over different similarity thresholds for SAST and DAST, respectively. We see that the F-scores increase with increasing similarity threshold, peaking at a threshold value ≥ 0.6 for DAST and at around 0.9 for SAST. Figure 3 in the appendix shows the performance metrics for clustering with knowledge graph-based semantic similarity. It is noteworthy that the F-scores for the knowledge graph-based clustering

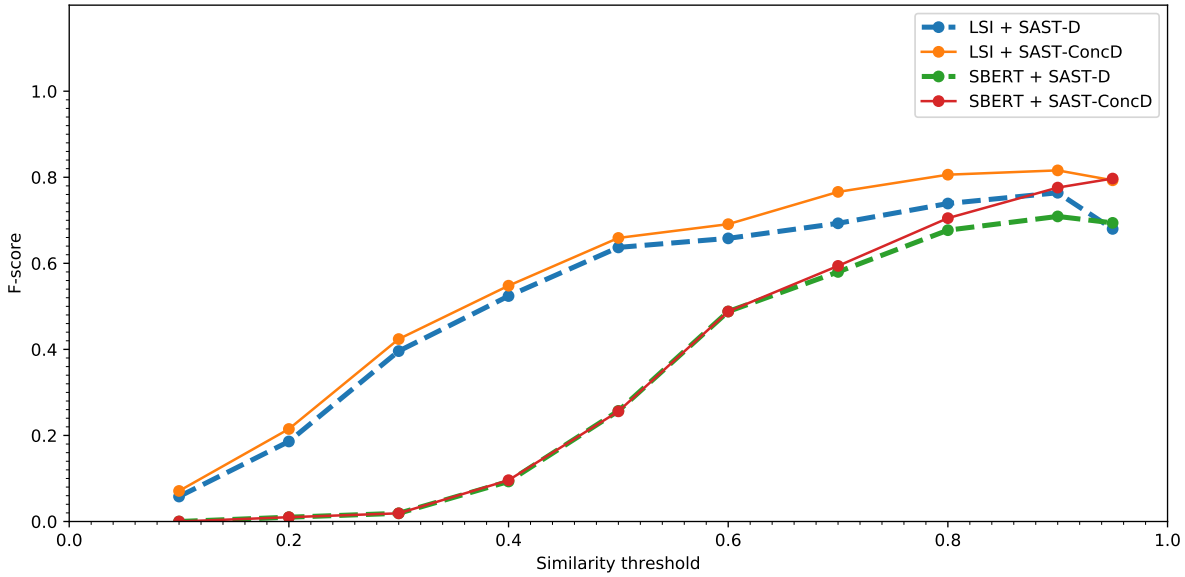


Figure 1: Semantic clustering results of SAST findings for different similarity thresholds.

are not only lower in comparison to *LSI* and *SBERT* but they also reach a plateau for threshold values higher than 0.2.

4.2.2 Qualitative Evaluation

For the qualitative evaluation, we showed incorrect predictions from the best results of semantic clustering of SAST and DAST findings to a security professional. The cluster results came from applying *LSI* to *SAST-ConcD* corpus for the SAST dataset and applying *SBERT* to *DAST-NDS* corpus for the DAST dataset. Using *SeFiLa*, the security professional inspected incorrect predictions and their associated ground-truth cluster. The security professional assigned possible reasons for poor duplicate identification by reading finding strings associated with incorrect predictions. These reasons are documented for 72 incorrect SAST predictions and 2 incorrect DAST predictions. The reasons and the number of times they were assigned to an incorrect prediction from either SAST or DAST clusters are listed in Table 4.

5 Discussion

From the quantitative evaluation, we see that SAST findings are best clustered by applying *LSI* to the *SAST-ConcD* corpus, which gives a F-score of 0.816. Although applying *SBERT* to the same corpora provides a similar F-score of 0.797 and matches a higher ratio of ground-truth clusters due to higher recall, *LSI* has a higher precision and less false positive predictions, which is a crucial requirement to the security professionals. Hence, applying *LSI* to *SAST-ConcD* corpus is our recommendation for identifying duplicate SAST findings.

When clustering DAST findings, we see that the highest F-score of 0.857 is achieved by applying *SBERT* and *LSI* to both *DAST-D* and *DAST-NDS* corpora. However, as illustrated in Figure 2, applying *SBERT* yields a high F-score for similarity threshold ≥ 0.6, whereas *LSI* yields a lower F-score. Since higher similarity thresholds are preferred in production scenarios to prevent false positive predictions, *SBERT* is preferred over *LSI*. For the corpus, *DAST-NDS* is preferred over *DAST-D* due to more textual content from three features, which leads to a better grasping of semantics and provides better distinction amongst false positives. We also see that for similarity threshold > 0.9, the F-score of *SBERT* with *DAST-NDS* slightly decreases. This is because of the strict distinction by semantic similarity algorithms, which also consider the semantics of a problem's solution when distinguishing between problems identified by different findings.

From the qualitative evaluation, we see that a significant challenge for SAST findings is the content of the finding description. Some tools provide a title instead of an actual description of the underlying problem. This leads to insufficient semantic content being derived from the finding corpus texts, thereby leading to poor duplicate identification. Another frequent reason for incorrect predictions in SAST are suboptimally constructed finding strings.

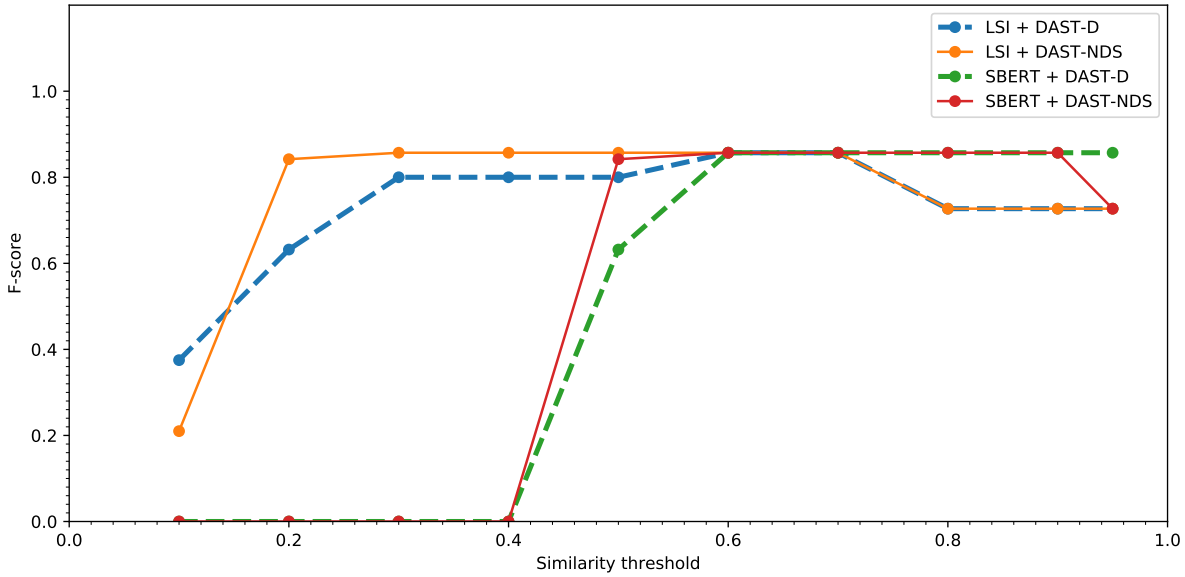


Figure 2: Semantic clustering results of DAST findings for different similarity thresholds.

This primarily arises when the information content of the original finding is low, which even is challenging for security professionals when determining duplicate findings just from reading the findings string. The third most highlighted challenge for SAST is the imbalance in the verbosity of description features of findings. The description either contains contextually rich problem descriptions or under-specified ones, leading to different extents of semantic content being captured and, thereby, incorrect comparisons being made. For DAST, we only have two incorrect predictions, which can be traced back to the challenge of being very application- and domain-specific.

To improve SAST findings clustering results, it is evident that the semantic content of finding strings representing a problem must be improved. This can be done by using multiple external sources, e.g., the National Vulnerability Database[4], the GitHub Advisory Database[5] for enrichment, or by scraping information from the multiple reference sources listed in a finding. The goal is that the textual data of each finding consists of multiple paragraphs and contains enough semantic content for the semantic similarity techniques to grasp as much contextual information as possible. Furthermore, the final corpus texts should contain the same verbosity level to avoid a bias related to the text length. Lastly, the same clustering approach can be studied using NLP models that are fine-tuned for security findings, accounting for the domain-specific vocabulary to improve the clustering results.

6 Limitations

While we present a variety of results regarding semantic clustering of security findings, our conclusions are limited in certain aspects. Firstly, all our findings result from scanning a single web application: JuiceShop. While it contains vulnerabilities encountered in real-world applications, it is restricted in its representation of a real scenario because JuiceShop is intended to comprise multiple vulnerabilities. Moreover, the subset of JuiceShop vulnerabilities that are clustered poorly might appear most often in reality, threatening the external validity of the results. Furthermore, our findings result from a finite number of modern security tools. While these tools are open-source and currently widely used, the scanning functionality of security testing tools is constantly evolving. Thereby, the scanning tools we use might change based on the needs of the domain. Lastly, our datasets were labeled by two security professionals and the results were evaluated by one security professional. While this is beneficial to prevent inconsistencies due to the subjective nature of the annotation tasks, the relevance of our results is highly dependent on the created ground-truth dataset. However, our chosen research design aims at making the results of our work as objective as possible. Researchers and

[4]https://nvd.nist.gov/
[5]https://github.com/advisories

Reason	Explanation for Incorrect Clustering	SAST	DAST
1	In the context of the product, this result can only be identified by somebody knowing the context of the application.	-	2
2	Different tools use a different phrasing to explain the same issue.	5	-
3	The tools sometimes provide no description of the finding. Hence, the features could only rely on the title.	39	-
4	Some tools provide more and some tools provide less text in their description, which reduces the impact of actual relevant features.	19	-
5	Additional review necessary due to an unknown reason for the decision.	5	-
6	The sub-optimally constructed feature string could be the reason for the incorrect clustering.	39	-
7	The tool describes the finding precisely according to the location of occurrence. Hence the finding text is over-specified.	3	-
8	Human annotation error and the suggested clustering by the algorithm is correct.	1	-
9	One tool addresses the issue of using an *eval* function, while the other one has the problem of user controlled values in it. However, it would not be considered as a major false positive.	3	-

Table 4: Overview of provided explanations from the qualitative evaluation.

practitioners can also use our developed annotation tool to reproduce our data collection or transfer our study insights to a setting of their own choice.

7 Conclusions and Future Work

In this work, we explored the applicability of semantic clustering of security findings through various similarity techniques. We tested three techniques from neural network-based, corpus-based, and knowledge-based methods on finding strings that describe security vulnerabilities identified by testing tools.

To this end, we created a ground-truth dataset of security findings clustered according to the expertise of security professionals. We compared this dataset to the results of semantic similarity techniques, indicating that SAST findings are best clustered by applying *LSI* to *SAST-ConcD* corpus, whereas DAST findings are best clustered by applying *SBERT* to *DAST-NDS* corpus. Conducting a qualitative evaluation with a security professional, we additionally pointed out the challenges encountered by semantic similarity techniques when applied to security findings and discussed possible solution strategies.

One potential future work would be the application of the chosen techniques to cluster security findings according to other testing strategies like solution-based clustering. This could grant deeper insights into the challenges of grouping security findings with NLP and provide access to new use cases. Furthermore, research on how plain neural networks perform when trained directly on semi-structured security findings appears to be promising given modern advancements in neural network architectures. Especially when compared to the NLP-based approach in this work, the properties of neural networks are worth exploring. Since neural networks automatically prioritize important features with layers like max-pooling, the manual effort undertaken to determine problem-describing features and clustering based on them could be alleviated. However, training a neural network requires significantly more data, so the construction of a much larger findings dataset would be necessary. Finally, an evaluation of the identified techniques in real-world DevOps scenarios could provide valuable insights into the practical usefulness of our approach in software development projects.

Acknowledgements

The authors want to thank the industry professionals for their exceptional effort in clustering the security findings and evaluating the shortcomings.

References

Karan Aggarwal, Finbarr Timbers, Tanner Rutgers, Abram Hindle, Eleni Stroulia, and Russell Greiner. 2017. Detecting duplicate bug reports with software engineering domain knowledge. *Journal of Software: Evolution and Process*, 29(3).

Dina Demner-Fushman and Jimmy Lin. 2006. Answer extraction, semantic clustering, and extractive summarization for clinical question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 841–848, Sydney, Australia. Association for Computational Linguistics.

Lee R Dice. 1945. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

Hamzeh Eyal Salman, Mustafa Hammad, Abdelhak-Djamel Seriai, and Ahed Al-Sbou. 2018. Semantic clustering of functional requirements using agglomerative hierarchical clustering. *Information*, 9(9).

Mohammad Hossin and Md Nasir Sulaiman. 2015. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1.

Adrian Kuhn, Stéphane Ducasse, and Tudor Gîrba. 2007. Semantic clustering: Identifying topics in source code. *Information and Software Technology*, 49(3):230–243. 12th Working Conference on Reverse Engineering.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240.

Olga Majewska, Diana McCarthy, Ivan Vulić, and Anna Korhonen. 2018. Acquiring verb classes through bottom-up semantic verb clustering. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

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*. Association for Computational Linguistics.

Phillip Schneider. 2020. App ecosystem out of balance: An empirical analysis of update interdependence between operating system and application software. Frankfurt University Library Johann C. Senckenberg.

Stacy Simpson. 2014. SAFECode whitepaper: Fundamental practices for secure software development 2nd edition. In *ISSE 2014 Securing Electronic Business Processes*, pages 1–32. Springer Fachmedien Wiesbaden.

Xi Tan, Xin Peng, Sen Pan, and Wenyun Zhao. 2011. Assessing software quality by program clustering and defect prediction. In *2011 18th Working Conference on Reverse Engineering*, pages 244–248.

A Supplementary Material

In this appendix, we provide additional material to the main article. Table 5 lists the security testing tools that were used to scan the web application JuiceShop and generate security findings. Figure 3 shows the performance metrics for clustering with knowledge graph-based semantic similarity.

Tool	Category	Analysis Type	Link
Anchore	SAST	Third-party vulnerabilities	anchore.com/opensource
Dependency Checker	SAST	Third-party vulnerabilities	owasp.org/dependency-check
Trivy	SAST	Third-party vulnerabilities	github.com/aquasecurity/trivy
GitLeaks	SAST	Hardcoded secrets	github.com/zricethezav/gitleaks
CodeQL	SAST	Coding flaws	codeql.github.com
Horusec	SAST	Coding flaws	horusec.io/site
Semgrep	SAST	Coding flaws	semgrep.dev
Arachni	DAST	Web app scan	github.com/Arachni/arachni
ZAP	DAST	Web app scan	www.zaproxy.org

Table 5: Overview of static (SAST) and dynamic (DAST) analysis security tools that were used to scan JuiceShop.

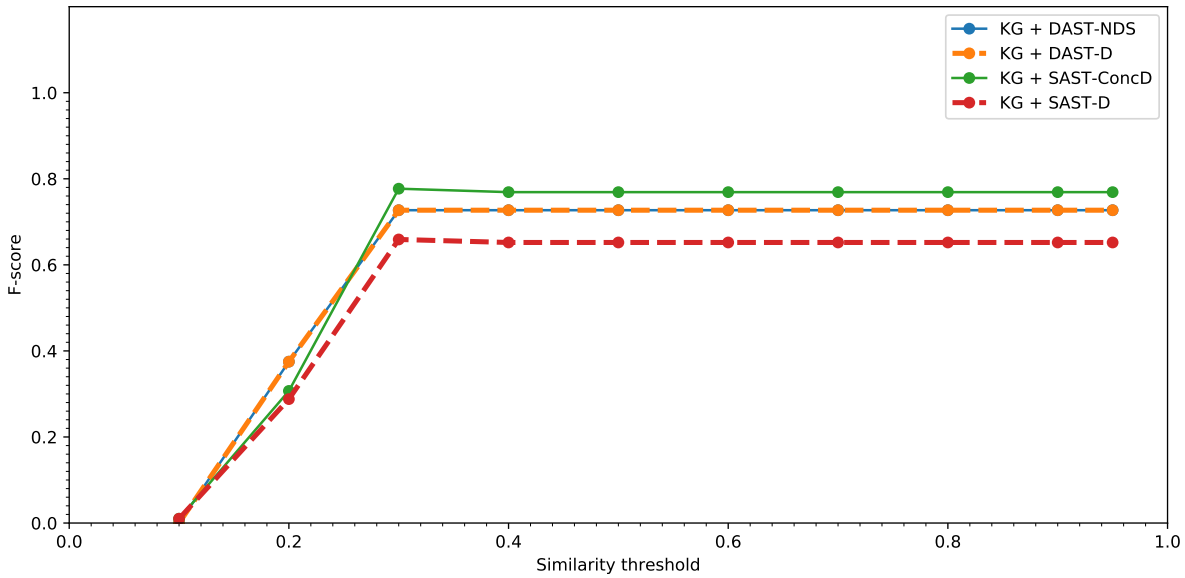


Figure 3: Semantic clustering results with knowledge graph-based similarity.

Contextual Embeddings Can Distinguish Homonymy from Polysemy in a Human-Like Way

Kyra Wilson [1] and **Alec Marantz** [1,2]
[1] Research Institute, New York University Abu Dhabi
[2] Departments of Psychology and Linguistics, New York University
{kyra.wilson, marantz} @ nyu.edu

Abstract

Lexical ambiguity is a pervasive feature of natural language, and a major difficulty in understanding language is selecting the intended meaning when more than one are possible. Despite this difficulty, many studies of single word recognition have found a processing advantage for ambiguous words compared to unambiguous ones. This effect is not homogeneous however–studies find consistent advantages for polysemes (words with multiple related meanings), and inconsistent results for homonyms (words with multiple unrelated meanings). Complicating this is the fact that most measures of ambiguity are derived from human- annotated or curated lexicographic resources, and their use is not consistent between studies. Our work investigates whether contextualized word embeddings are able to capture human-like distinctions between senses and meanings, and whether they can predict human behavior. We reanalyze data from previous experiments reporting ambiguity (dis)advantages using the lexical decision times reported in the English Lexicon Project. We find that our method does replicate the polyseme advantage and homonym disadvantage previously reported, and the predictors are superior to binary distinctions derived from lexicographic resources. Our findings point towards the benefits of using continuous-space representations of senses and meanings over more traditional measures. Additionally, we make our code publicly available for use in future research.

1 Introduction

Distributed representations of meaning (word embeddings) have brought great advancements to many natural language processing tasks including sentiment analysis, text summarization, and translation, to name just a few. Outside of computational applications, these embeddings have also been used successfully in psycholinguisitcs to predict semantic priming data (Ettinger and Linzen, 2016), eye-tracking data (Søgaard, 2016), and even neural activations (Honari-Jahromi et al., 2021). Despite their success in a wide variety of fields and tasks, these static representations' performance is greatly limited because each orthographic wordform is limited to only one vector representation.

This is problematic because ambiguity is quite pervasive in natural language. Words that have the same spelling can have multiple senses (polysemes) and/or meanings (homonyms). For example, in Wordsmyth online dictionary, *slam* has two entries (meanings). Under the first entry there are multiple senses–a noun meaning "sharp criticism", a verb meaning "shut something loudly", an additional noun meaning "the sound made by shutting something loudly", and others. Under the second there are additional senses unrelated to the first entry's senses–a noun meaning "winning of all tricks in a card game" and another noun meaning "a poetry reading event." Despite all these different usages, *slam* has just a single unchanging spelling and pronunciation.

To successfully use language, both humans and language models must somehow be able to select a single meaning from a set of multiple candidates for ambiguous words. For models based on static representations, this was not a straightforward task because all of the possible senses and meanings were collapsed into a single representation that was used invariantly across any possible context. There were no senses or meanings for the model to choose among. This flaw impacts a model's ability to understand the true meaning of words when they are used in changing contexts.

One recent advancement to address this problem is the use of contextualized word embeddings such as ELMo and BERT. Instead of having a single representation per wordform, these systems produce embeddings that change dynamically based on the surrounding context of a single word occurrence.

This way, *slam* used in the sentences *When I'm mad I slam the door* and *I attended the poetry slam last night* will have two distinct representations despite sharing the same orthographic form. The use of these contextualized embeddings have provided even further progress in a wide variety of computational applications because they successfully address the ambiguity problem.

While contextualized embeddings and transformer architectures have begun to be adopted in the analysis of human language processing (Jain and Huth, 2018; Kumar et al., 2022; Heilbron et al., 2021), this body of work has largely focused on language processing in context, rather than at the single-word level. In this work, we attempt to show that contextual embeddings can also be useful for analyzing human language processing even in the absence of context by looking specifically at the "ambiguity advantage". This is a widely studied psycholinguistic phenomenon in which ambiguous words are recognized faster than unambiguous ones in lexical decision experiments[1].

In previous work investigating the ambiguity advantage, senses and meanings have often been distinguished based on how they are listed in lexicographic resources (Rodd et al., 2002). Meanings generally correspond to dictionary entries, while senses will correspond to the various distinguished uses within those entries. (Following the previous example, *slam* would have two meanings, and at least five senses). Additionally, senses are generally assumed to be related and share some semantic core between them, while meanings have no shared semantics and are unrelated. For example, two of *slam*'s senses both have something to do with shutting something and making a noise–one is the action and one is the resulting sound; clearly there is a shared semantic core here. However, it is not clear that there is a semantic core shared between these senses and the senses of the other meaning, such as winning tricks in a card game.

Even though lexicographic resources do distinguish senses and meanings, using them to study the ambiguity advantage is challenging because they typically lack explicit criteria or explanations to why particular distinctions of relatedness are made. For example, a *door slam* and a *slam of the production* are considered related to each other according to Wordmsyth online dictionary (even though the latter doesn't necessarily have any meaning related to shutting something or a noise), but neither are related to a *poetry slam*.

Understandably, extensive discussions of the nature of semantic relatedness is typically outside the scope of most lexicographic resources, but this means they are not very well-suited to psycholinguistic research where such distinctions are of great importance. For this purpose, a more useful measure of a word's senses and meanings would be derived from the way speakers use the words at present as opposed to lexicographer categorizations and would have clear criteria for what makes something a sense versus a separate meaning.

In this study, we used BERT to derive a new measure of a word's numbers of senses and meanings [2], and we apply this measure to previously gathered lexical decision data. We compare our results to those from a previous study which quantified ambiguity using lexicographic resources and find that ours perform at least as well. This points to the benefits of further adopting contextualized embeddings for use in psycholinguistic research.

2 Related Work

Comparing the way that ambiguous and unambiguous words are processed can give information about the organization of the mental lexicon and ways in which different kinds of words may be retrieved and recognized. This is primarily tested in lexical decision experiments, where a mix of target stimuli and non-words are presented one at a time, and participants respond as quickly as possible with whether or not they recognize the presented string. Their reaction times on the target stimuli are then analyzed to determine what variables make word recognition easier (faster response times) or harder (slower responses times). These experiments generally reveal that there is, in fact, a difference in the way words with multiple senses and/or meanings are processed as compared to unambiguous words.

Most studies find that multiple senses facilitate recognition, as evidenced by a faster reaction time for words with multiple senses in a lexical decision paradigm (Borowsky and Masson, 1996; Hino and Lupker, 1996) compared to unambiguous words.

[1]This finding has been observed in both single word recognition tasks (Rodd et al., 2002; Borowsky and Masson, 1996; Hino and Lupker, 1996) and sentence presentation contexts (Frazier and Rayner, 1990; Klepousniotou, 2002). In this work, we focus exclusively on advantages for single word recognition.

[2]The tools developed for this study are available at https://github.com/kyrawilson/word-senses-from-CWE.

This lends support to a model of word recognition where words with multiple senses have multiple semantic representations, leading to easier recognition as a result of increased semantic activation compared to words with fewer senses.

However, the same result is not as consistent for words with multiple meanings. Some studies find increased reaction time compared to unambiguous words (Rodd et al., 2002; Beretta et al., 2005) (suggesting that having multiple unrelated meanings may make recognition more difficult because of competing activations), while others find an equivalent advantage for both multiple senses and multiple meanings (Hino et al., 2010; Pexman et al., 2004). Because the results are mixed, it is unclear whether words with multiple meanings are stored and accessed in a way similar to those with multiple senses, or whether they are different in some critical way.

It has been proposed that the contradictory results for words with multiple meanings are a consequence of differing methodologies in selecting ambiguous stimuli (Haro and Ferré, 2018). Namely, experimenters use a variety of sources for selecting ambiguous words because there is no gold standard resource for differentiating between related senses and related meanings; thus differences arise not only in what sources are used, but also in what the individual sources classify as ambiguous words since they are curated by different groups using varied techniques. This paper shows that that with advances in distributed representations of meanings, previous measures that relied on lexicographic sources can be exchanged for measures derived from contextual representations (specifically contextualized meaning vectors from BERT, a transformer-based language model) without reference to any outside resources, and these measures will perform at least as well as traditional ones.

There have been previous attempts to identify information about word senses from BERT embeddings. Reif et al. (2019) sampled sentences from Wikipedia and found that similar contextual usages of words tended to cluster together in meaning vector space and that the spatial location of a word could be changed by altering the context sentence. This suggests that BERT is able to represent meaningful semantic information within a subset of its vector dimensions [3]. Following Reif et al. (2019), there have been multiple attempts to use BERT for word sense disambiguation, including some which also use lexicographic resources to interpret the disambiguated senses (Wiedemann et al., 2019; Du et al., 2019; Vial et al., 2019).

In addition, there has also been research investigating how BERT represents words with different numbers of senses and meanings. Garí Soler and Apidianaki (2021) investigated both whether BERT could distinguish words with a single versus multiple senses and whether the senses cluster in interpretable ways. First, they found that usages of words with a single sense (according to WordNet) had a higher similarity than words with multiple senses. Furthermore, they used a k-means algorithm to cluster senses of ambiguous words, and they found that the quality of this clustering was high and correlated with annotator judgements about sense similarities. Although this study did demonstrate the potential of using clustering to analyze BERT embeddings, the use of the k-means algorithm is suboptimal because the number of clusters must be known a priori, and thus does not extend well to applications in which human annotations are unavailable or contradictory.

There has also been work investigating how BERT's representations of polysemy may correspond to humans'. Nair et al. (2020) collected human judgements of meaning relatedness for homonyms and polysemes and compared them to distances in BERT embedding space. They found that homonym meanings were more reliably distant than related senses. This suggests that the way BERT represents information is somewhat consistent with human intuitions. However, the experimental task in this study was metalinguistic: people were asked about how they use language, which may or may not be consistent with actual language use.

An additional test of how well BERT corresponds with human language would be to use it to predict actual human behavior rather than intuitions. Therefore in our study, we explore BERT's similarities to human language knowledge, analyzing behavioral reaction time data to potentially ambiguous and polysemous words and correlating human reaction times to the numbers of senses and meanings derived from BERT embeddings.

[3] A similar result was observed by Thompson and Mimno (2020) in the topic modeling domain.

3 Methods

3.1 Data

For the 182 words (124 ambiguous and 58 unambiguous) used in the first experiment of Rodd et al. (2002), we retrieved their mean reaction time in a visual lexical decision experiment from the English Lexicon Project (ELP) (Balota et al., 2007). These reaction times were used as the response variable in a linear regression model.

The words used in this experiment were selected by Rodd et al. (2002) to amplify the differences between ambiguous and unambiguous words. Of the 124 ambiguous words, 113 were taken from the Twilley et al. (1994) homograph norms, and the remaining 11 were judged to have similar properties. Most of these words were judged to have two or three meanings according to the original annotations, where meanings and senses were conflated, and half of them had two distinct entries in the Wordsmyth dictionary (corresponding to two meanings). The other half only had a single entry; the other "meaning" was annotated by Wordsmyth as a sense instead. This difference in the two groups allowed for a comparison of meaning relatedness. The words with two Wordsmyth entries were considered ambiguous (homonyms) while the remaining 58 words in the stimuli set were identified as being unambiguous (polysemes, since they were judged to have multiple senses) and had only one meaning.

We also included a number of control variables in our analysis in line with Rodd et al. (2002), including log word frequency, length, orthographic neighborhood, and concreteness. These were also collected from ELP.

3.2 Number of Senses

Our method for deriving the number of senses for a word assumes that same senses will be used in similar contexts, and therefore the contextual embeddings for a word in a particular sense will also be similar to each other. Furthermore, other senses will have dissimilar enough contexts that we can derive a measure of the number of senses by applying a clustering algorithm (HDBSCAN) to the BERT embeddings, where the identified clusters will correspond to individual senses of a word.

HDBSCAN (Campello et al., 2013) is a hierarchical clustering algorithm which uses the stability and persistence of clusters in order to select an optimal clustering from the hierarchy. It works by first identifying areas of high and low density points and deriving a distance (mutual reachability) metric that amplifies the distance to sparse points. Next, a minimum spanning tree is constructed using the mutual reachability distance and then converted into a hierarchy by sorting the edges in increasing order and creating a new cluster for each edge. Finally, a single clustering is selected from the hierarchy by selecting the clusters with the greatest stability, meaning that for a large range of distance values the cluster remains as a whole and does not split into two smaller clusters.

The use of HDBSCAN is particularly suited to the clustering of word senses for two reasons [4]. First, the algorithm allows extreme outlier points to be categorized as noise rather than coercing them into a cluster. This is good for our application because of the flexibility of language. Even though words have a generally standard and accepted set of meanings, there is nothing to prevent novel usages of a word in a new context. For our purposes, we would like to avoid including very low-frequency senses or meanings which are unlikely to be known by a majority of speakers.

Additionally, the only hyperparameter of the algorithm is the minimum number of points a cluster must contain, in contrast to other clustering algorithms in which the number of clusters must be specified a priori. We are interested in deriving the number of different senses from an unlabelled corpus rather than simply identifying the sense clusters which correspond to entries in lexicographic resources. Another side effect of this is that we are able to specify how many usages a particular sense must have in order to be considered well-known and avoid contaminating our clusters with too many "noise usages". We specified that our clusters should contain, at minimum, at least one percent of the points in the total number of embeddings for a given word.

Following Reif et al. (2019), we first sampled 1,000 occurrences of each word in Rodd et al. (2002)'s stimuli set from English Wikipedia[5], and used the publicly available pre-trained $\text{BERT}_{\text{BASE}}$ model (Devlin et al., 2019) in combination with the Hugging Face (Wolf et al., 2020) and Flair li-

[4] A related algorithm, DBSCAN (Ester et al., 1996), has also been shown to have success in clustering word embeddings (Mohammed et al., 2020). We chose to use HDBSCAN due to its increased flexibility over DBSCAN.

[5] For one word (*poach*), there were only 578 occurrences in Wikipedia. We used all of the occurrences in this case.

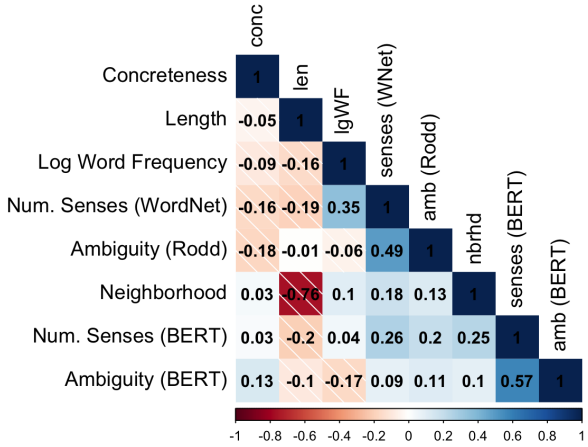


Figure 1: Spearman's rank correlation between all predictors.

braries (Akbik et al., 2019) to encode the word in their context sentences[6]. The word token of interest was then extracted from each context sentence and its layers were averaged, resulting in a single 768-dimension embedding for each sentence [7].

Finally, the embedding dimensions were reduced from 768 to two using t-SNE (Van der Maaten and Hinton, 2008). HDBSCAN is not guaranteed to perform well for high-dimensional data, so we chose to have it operate over embeddings that were also used for visualization in order to aid with interpretability of the clustering results. For each word, the minimum cluster size was one percent of the total number of embeddings for that word.

3.3 Ambiguity

Since there can be multiple senses of a word within a single meaning, we were interested in identifying any superstructure amongst the clusters which might correspond to different meanings. Broadly, to identify meanings, we are now aiming to cluster the senses of words themselves rather than the individual usages as an attempt to join senses that are most similar to each other. We do this by only clustering a subset of the points used in the the number of senses calculation as well as increasing the minimum cluster size hyperparameter in the HDBSCAN algorithm. This way, we are able to use the same algorithmic approach to derive unique measures for number of senses and ambiguity.

To begin, we select a subset of points to use for identifying meaning clusters. This is done in order to make the data sufficiently different to avoid recreating identical senses clusters as well as eliminating possible noise usages from the meaning clustering. The subset of points we used were those identified as "exemplars" by HDBSCAN within each of the identified sense clusters. In this implementation, exemplar points are those which persist in their cluster for the largest range of distance values and which are generally centrally located in their respective clusters. In other words, the exemplars are the points which are identified as being the strongest members of the cluster and least likely to be noise.

After identifying the set of exemplar points for each cluster, we used HDBSCAN clustering again in order to identify any potential higher order clusters. In contrast to the number of senses clustering, in this iteration we allowed the clustering algorithm to assign all the exemplar points to a single cluster, under the assumption that some subset of the stimuli are unambiguous and should thus have only one meaning.

Another difference between the ambiguity clustering and the number of senses clustering is the minimum cluster size. It has been observed that there is interpretable structure even within sense clusters (Reif et al., 2019). For example, for the word *die*, Reif and colleagues found that within a single sense cluster there was a separation relating to the number of people who died. We wanted to avoid the formation of even more granular sense clusters, so in this iteration we set the minimum cluster size to be the size of the smallest set of exemplar points from a single sense cluster. Finally, if the clustering procedure still resulted in a larger value for ambiguity (number of meanings) than the number of senses, we assigned the number of meanings to be equal to the number of senses post-hoc.

4 Results

4.1 Qualitative Analysis

An example of the clustering of senses and meanings can be seen in Figure 2 for the word *tent*, which has three senses and one meaning according to our proposed method. The three different shapes indicate that there were three senses identified–one that has to do with *tent* as a physical object used

[6]We only selected from sentences in which the target word appeared a single time.

[7]Multiple studies have shown that semantic representations differ depending on the BERT layer (Garí Soler and Apidianaki, 2021; Jawahar et al., 2019). While we averaged all layers together, it is possible that selecting a single layer would yield higher performance. We leave this investigation for future work.

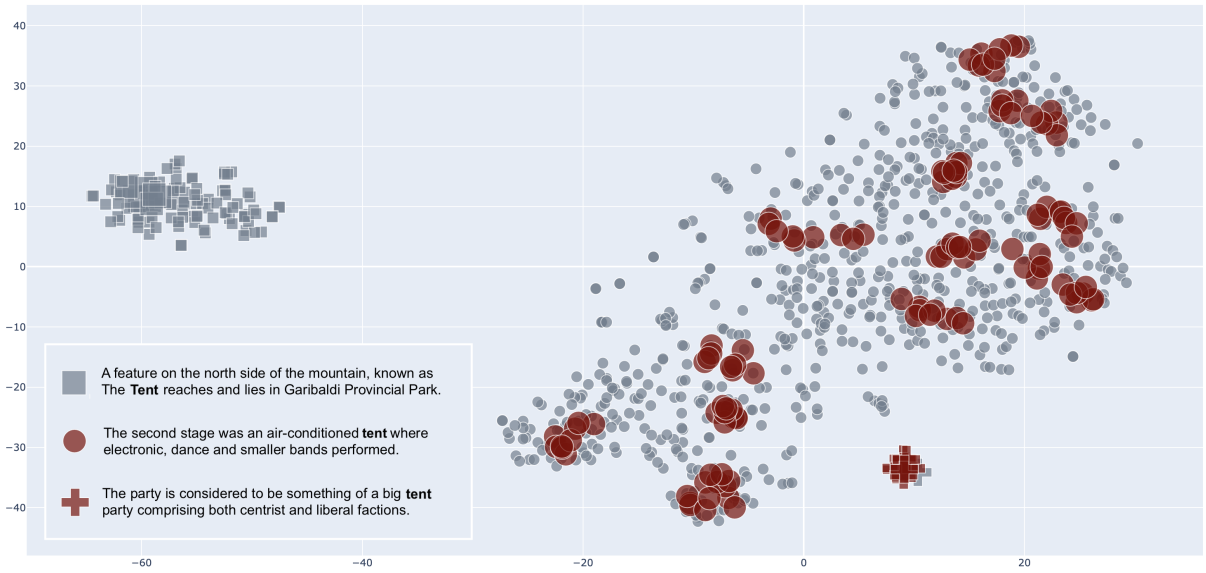


Figure 2: Example of sense and meaning clustering for *tent*, including example usages in sentences from Wikipedia. Different senses are indicated by different shapes, exemplar points used to cluster meanings are larger, and colored points indicate meaning groups.

for shelter, one that is a part of the phrase *big tent party*, and a third where *tent* is part of a title or used as a proper noun. The senses are accurately separated into groups that have internal cohesion, but separated from other groups with slightly different semantics.

The single group of red points indicate that *tent* has only one meaning combining the "physical object" and "political party" uses. Although these senses are not interchangeable, they are clearly related. Just as people might congregate under tents at a concert, they also metaphorically congregate in a *big tent party*. The third sense, however, is both unrelated to this meaning and also not cohesive enough to form its own separate meaning. The cluster contains titles and other proper nouns usages, so *tent* is both unlikely to have a single shared context corresponding to a new meaning in this cluster or a context close enough to the other two senses that it should be included in the first meaning. Therefore, it is correctly identified as "noisy" usages of *tent* and not analyzed as an additional meaning.

4.2 Number of Senses

To begin, we compared the BERT-derived number of senses to the number of senses as indicated in WordNet[8]. There was a weak positive correlation between the BERT-derived number of senses and the number of senses reported by WordNet (ρ = 0.26), as shown in Figure 1. We entered both predictors into a linear regression model with response time in a lexical decision task as reported in the ELP as the dependent variable. The full model results are shown in Figure 3.

Only the number of senses as derived from BERT was a significant predictor of reaction time, and the effect replicated what has been reported in previous studies. Words with more senses were generally recognized faster than those with fewer senses. This effect can be seen in Figure 4. Next we performed an ANOVA to assess whether additional variance is explained by our predictor. As expected, the ANOVA indicated that including the number of senses derived from the contextual embeddings did improve the model fit (F = 3.78, p = 0.05).

4.3 Ambiguity

We compared the binary ambiguity variable used by Rodd et al. (2002) with our continuous variable derived from contextualized embeddings. There was low correlation between the binary ambiguity variable and our BERT-derived variable (ρ = 0.11). In the model with none of our predictors, we did not replicate the ambiguity effect reported by Rodd et al. (2002). In fact, we found the opposite; Rodd et al. (2002) reported an inhibitory effect where ambiguous words were recognized more slowly than unambiguous words, but our analysis showed that ambiguity made reaction time faster (just as

[8]The Rodd et al. (2002) study did not indicate how the number of senses was calculated, so we used WordNet as an approximation of their metric.

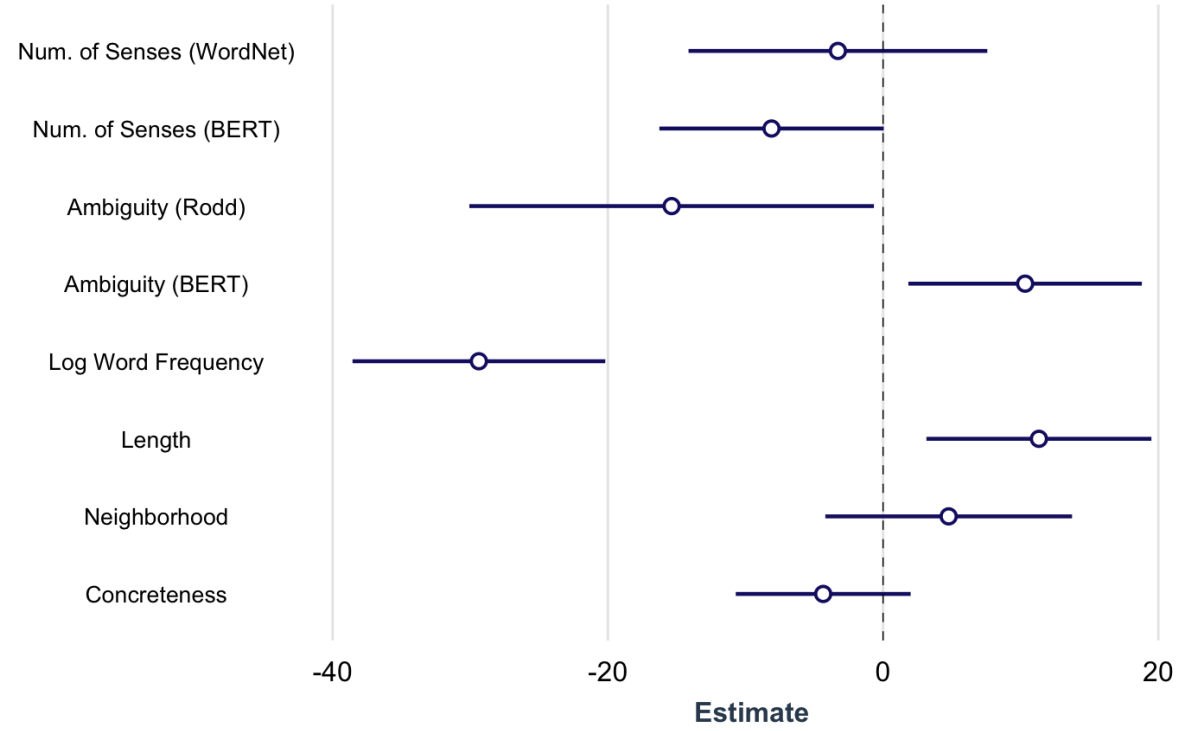


Figure 3: Estimates of linear regression coefficients predicting reaction time. The significant predictors are the number of senses and meanings derived from contextual embeddings, the binary ambiguity variable from Rodd et al. (2002), log word frequency, and length.

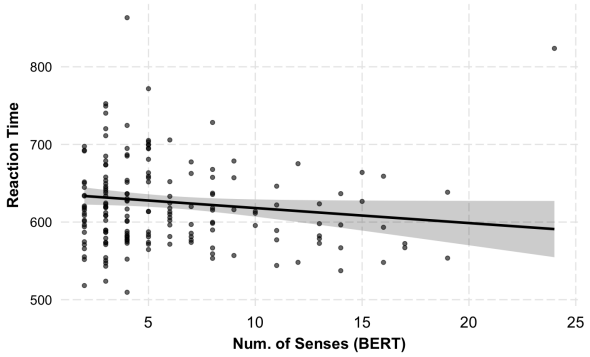


Figure 4: Regression line showing inverse relationship between number of senses and reaction time.

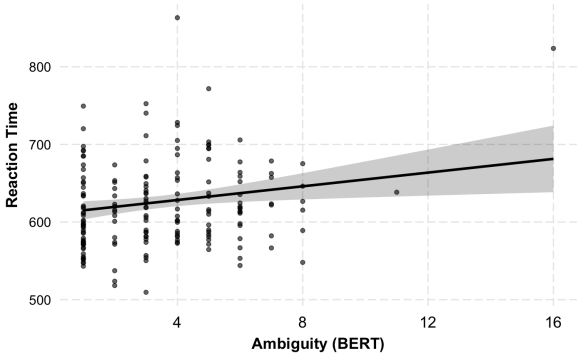


Figure 5: Regression line showing direct relationship between ambiguity and reaction time.

multiple senses facilitate recognition).

However, when we included our predictors, we found that the ambiguity variable as derived from BERT did produce an inhibitory effect as originally reported, as shown in Figure 5. Compring models with and without the ambiguity in ANOVA showed that our predictors also significantly improved the fit of the model ($F = 6.61$, $p = 0.01$).

5 Discussion

There are multiple important results from this investigation. First, we found that the contextual embeddings not only correspond to human judgements as previously reported (Nair et al., 2020), but also to human behavior. Our number of senses measure replicated the well-reported finding that having multiple senses is facilitatory in word recognition (more senses lead to faster identification). In fact, for this particular set of stimuli, our measure outperformed the more traditional measure derived from WordNet in predicting reaction times in a lexical decision task.

The results for ambiguity (number of meanings) are slightly more complex. First, our analysis did

not replicate the original results when using the binary ambiguous/unambiguous variable as computed by Rodd et al. (2002). We instead found an additional facilitatory effect for this variable where multiple meanings correspond to faster recognition as compared to single meanings. However, our number of meanings variable, derived from clustering senses using sense exemplars, did result in words with more meanings having slower reaction times as previously reported, above and beyond the effects of the binary variable. For theories of word recognition, it is not immediately apparent why these two variables should have opposite effects, but as our measure has consistent criteria and clear definitions for deriving predictor values, further experiments should be able to investigate this in depth using a wider variety of stimuli.

Finally, another interesting outcome worth further investigation is that our results were obtained using only two-dimensional embeddings derived from BERT. Previous experiments investigating the representations of polysemy and ambiguity within BERT have done so using all 768 dimensions of the embeddings (Reif et al., 2019; Garí Soler and Apidianaki, 2021), while our experiment suggests similar information can be represented using far fewer dimensions. Determining the optimal number of dimensions for representing polysemy and ambiguity using BERT remains an open question worth further study.

The replication of the previous ambiguity advantage results show how contextual embeddings such as BERT can be useful in the analysis of experimental data. For the number of senses advantage, we showed a stronger effect than more traditional predictors relying on lexicographers. For the number of meanings, we also replicated previous findings and found that our predictor performed just as well as traditional ones. However, because our predictor was derived from unlabelled corpora without resorting to any human annotation (which may introduce bias) we find it methodologically superior to predictors derived from lexicographic resources such as dictionaries and WordNet. We think that continuing to use contextual embeddings to derive predictors will facilitate transparency and replicability across many different areas of linguistic research as well as allowing for more flexibility in what words and languages are able to be studied.

Finally, another potential benefit of this methodology is the possibilities of extending it to languages other than English. Generating high-quality lexicographic resources is very time- and labor-intensive, so current research into the ambiguity advantage is limited to those languages which already have such resources. Our methodology, on the other hand, could theoretically be extended to any language which has a pre-trained model able to produce contextual embeddings (or for a slightly higher cost, any language for which a new contextual embedding model could be trained and deployed), and further research should be done to verify that the properties of BERT embeddings observed in this experiment would also be present in models trained on other languages.

6 Conclusion

This study further supports work which indicates that contextualized embeddings contain information which is able to predict human language processing. We extended the approaches of earlier work by not only deriving a measure of how many senses a word has, but also finding how many distinct meanings a word has by clustering those senses. We used these numbers to replicate the finding that multiple senses facilitate recognition in a lexical decision experiment and add support to the finding that multiple meanings inhibit word recognition. This is an important result because it suggests this method can be used as a replacement for traditional ways of deriving measures of ambiguity and polysemy, allowing for standardization of variable predictors across experiments in order to facilitate comparison and minimize conflicting results.

Acknowledgements

This research was supported by the NYUAD Research Institute under Grant G1001 and was carried out on the High Performance Computing resources at New York University Abu Dhabi. We additionally thank Tal Linzen and anonymous reviewers for their guidance and feedback on earlier versions of this paper.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for*

Computational Linguistics (Demonstrations), pages 54–59.

David A Balota, Melvin J Yap, Keith A Hutchison, Michael J Cortese, Brett Kessler, Bjorn Loftis, James H Neely, Douglas L Nelson, Greg B Simpson, and Rebecca Treiman. 2007. The english lexicon project. *Behavior research methods*, 39(3):445–459.

Alan Beretta, Robert Fiorentino, and David Poeppel. 2005. The effects of homonymy and polysemy on lexical access: An MEG study. *Cognitive Brain Research*, 24(1):57–65.

Ron Borowsky and Michael EJ Masson. 1996. Semantic ambiguity effects in word identification. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22(1):63.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer.

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.

Jiaju Du, Fanchao Qi, and Maosong Sun. 2019. Using BERT for word sense disambiguation. *arXiv preprint arXiv:1909.08358*.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Allyson Ettinger and Tal Linzen. 2016. Evaluating vector space models using human semantic priming results. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 72–77, Berlin, Germany. Association for Computational Linguistics.

Lyn Frazier and Keith Rayner. 1990. Taking on semantic commitments: Processing multiple meanings vs. multiple senses. *Journal of Memory and Language*, 29(2):181–200.

Aina Garí Soler and Marianna Apidianaki. 2021. Let's Play Mono-Poly: BERT Can Reveal Words' Polysemy Level and Partitionability into Senses. *Transactions of the Association for Computational Linguistics*, 9:825–844.

Juan Haro and Pilar Ferré. 2018. Semantic ambiguity: Do multiple meanings inhibit or facilitate word recognition? *Journal of Psycholinguistic Research*, 47(3):679–698.

Micha Heilbron, Kristijan Armeni, Jan-Mathijs Schoffelen, Peter Hagoort, and Floris P de Lange. 2021. A hierarchy of linguistic predictions during natural language comprehension. *BioRxiv*, pages 2020–12.

Yasushi Hino, Yuu Kusunose, and Stephen J Lupker. 2010. The relatedness-of-meaning effect for ambiguous words in lexical-decision tasks: When does relatedness matter? *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 64(3):180.

Yasushi Hino and Stephen J Lupker. 1996. Effects of polysemy in lexical decision and naming: An alternative to lexical access accounts. *Journal of Experimental Psychology: Human Perception and Performance*, 22(6):1331.

Maryam Honari-Jahromi, Brea Chouinard, Esti Blanco-Elorrieta, Liina Pylkkänen, and Alona Fyshe. 2021. Neural representation of words within phrases: Temporal evolution of color-adjectives and object-nouns during simple composition. *PloS one*, 16(3):e0242754.

Shailee Jain and Alexander Huth. 2018. Incorporating context into language encoding models for fMRI. *Advances in Neural Information Processing Systems*, 31.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Ekaterini Klepousniotou. 2002. The processing of lexical ambiguity: Homonymy and polysemy in the mental lexicon. *Brain and Language*, 81(1-3):205–223.

Sreejan Kumar, Theodore R Sumers, Takateru Yamakoshi, Ariel Goldstein, Uri Hasson, Kenneth A Norman, Thomas L Griffiths, Robert D Hawkins, and Samuel A Nastase. 2022. Reconstructing the cascade of language processing in the brain using the internal computations of a transformer-based language model. *bioRxiv*.

Shapol M Mohammed, Karwan Jacksi, and Subhi RM Zeebaree. 2020. Glove word embedding and dbscan algorithms for semantic document clustering. In *2020 International Conference on Advanced Science and Engineering (ICOASE)*, pages 1–6. IEEE.

Sathvik Nair, Mahesh Srinivasan, and Stephan Meylan. 2020. Contextualized word embeddings encode aspects of human-like word sense knowledge. In *Proceedings of the Workshop on the Cognitive Aspects of the Lexicon*, pages 129–141, Online. Association for Computational Linguistics.

Penny M Pexman, Yasushi Hino, and Stephen J Lupker. 2004. Semantic ambiguity and the process of generating meaning from print. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 30(6):1252.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of BERT. *Advances in Neural Information Processing Systems*, 32.

Jennifer Rodd, Gareth Gaskell, and William Marslen-Wilson. 2002. Making sense of semantic ambiguity: Semantic competition in lexical access. *Journal of Memory and Language*, 46(2):245–266.

Anders Søgaard. 2016. Evaluating word embeddings with fMRI and eye-tracking. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 116–121, Berlin, Germany. Association for Computational Linguistics.

Laure Thompson and David Mimno. 2020. Topic modeling with contextualized word representation clusters. *arXiv preprint arXiv:2010.12626*.

Leslie C Twilley, Peter Dixon, Dean Taylor, and Karen Clark. 1994. University of alberta norms of relative meaning frequency for 566 homographs. *Memory & Cognition*, 22(1):111–126.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2019. Sense vocabulary compression through the semantic knowledge of WordNet for neural word sense disambiguation. In *Proceedings of the 10th Global Wordnet Conference*, pages 108–117, Wroclaw, Poland. Global Wordnet Association.

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does BERT make any sense? Interpretable word sense disambiguation with contextualized embeddings. *arXiv preprint arXiv:1909.10430*.

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.

A Polysemy and Homonymy Values

Word	Ambiguity (Rodd)	Senses (WordNet)	Meanings (BERT)	Senses (BERT)
admit	Amb.	8	1	2
advance	Amb.	20	8	15
affair	Amb.	3	1	14
alone	Unamb.	6	2	5
amuse	Unamb.	2	1	3
apple	Unamb.	2	3	3
arms	Amb.	10	8	11
article	Amb.	5	5	6
baby	Unamb.	8	2	3
badger	Amb.	4	3	3
bark	Amb.	9	3	3
batter	Amb.	5	4	4
Bible	Unamb.	2	1	3
blind	Amb.	10	2	14
bonnet	Amb.	3	2	3
bowl	Amb.	12	4	6
boxer	Amb.	4	1	4
brain	Unamb.	7	5	5
bridge	Amb.	12	1	8
broke	Amb.	60	1	17
brutal	Unamb.	4	1	2
bulb	Amb.	6	5	5
bus	Unamb.	7	7	8
cabinet	Amb.	4	1	3
calf	Amb.	4	4	4
can	Amb.	8	1	2
cane	Amb.	4	6	6
case	Amb.	22	1	3
cattle	Unamb.	1	3	3
chance	Amb.	9	5	5
charm	Amb.	8	4	4
chest	Amb.	4	3	3
China	Amb.	4	1	2
cider	Unamb.	1	3	3
cigar	Unamb.	1	3	3
citizen	Unamb.	1	5	5
clay	Unamb.	5	3	6
clog	Amb.	9	4	4
coal	Unamb.	5	3	3
company	Amb.	10	3	4
craft	Amb.	6	6	10
cricket	Amb.	3	1	8

Word	Ambiguity (Rodd)	Senses (WordNet)	Meanings (BERT)	Senses (BERT)
custard	Unamb.	1	3	3
deed	Amb.	2	3	5
degree	Amb.	7	7	9
dense	Amb.	4	1	4
destroy	Unamb.	4	1	2
diamond	Unamb.	6	6	10
digit	Amb.	3	6	6
dollar	Unamb.	4	1	3
dozen	Unamb.	2	1	2
dry	Amb.	19	1	7
express	Amb.	13	6	6
fee	Unamb.	3	2	3
feet	Amb.	11	5	5
fence	Amb.	7	1	2
firm	Amb.	14	3	3
fling	Amb.	7	1	2
forest	Unamb.	3	5	5
fraud	Unamb.	3	1	2
free	Amb.	22	1	7
frog	Unamb.	4	4	4
fun	Unamb.	4	3	4
glare	Amb.	6	3	3
glass	Amb.	12	4	4
glove	Unamb.	3	5	8
goat	Unamb.	4	4	4
grain	Amb.	15	2	3
grief	Unamb.	2	1	2
grow	Unamb.	10	1	2
hamper	Amb.	4	1	3
hill	Unamb.	6	2	7
horn	Amb.	12	6	8
hotel	Unamb.	1	4	5
interest	Amb.	10	2	2
item	Unamb.	6	1	2
jumper	Amb.	8	6	15
kid	Amb.	7	3	3
kind	Amb.	4	7	13
kingdom	Unamb.	6	1	2
lake	Unamb.	3	5	8
last	Amb.	21	1	12
late	Amb.	11	1	2
lean	Amb.	10	3	9
left	Amb.	24	1	3
letter	Amb.	8	1	13
lie	Amb.	10	6	14
like	Amb.	11	1	3
limp	Amb.	5	2	5
lobby	Amb.	4	4	5
lung	Unamb.	1	8	11
marble	Amb.	4	3	3
march	Amb.	14	6	10
maroon	Amb.	6	4	4
metal	Unamb.	4	4	4
might	Amb.	1	1	3
misery	Unamb.	2	4	5
nail	Amb.	10	6	11
net	Amb.	12	1	14
novel	Amb.	4	1	9
ocean	Unamb.	2	4	4
odd	Amb.	6	8	8
organ	Amb.	6	4	4
palm	Amb.	5	6	6
panel	Amb.	10	5	6
park	Amb.	8	5	9
patient	Amb.	3	3	4
peer	Amb.	3	11	19
picket	Amb.	8	5	5
pine	Amb.	3	4	4
pitcher	Amb.	5	6	13
poach	Amb.	2	16	24
poet	Unamb.	1	1	6
poker	Amb.	2	3	4
pole	Amb.	13	5	11
prayer	Unamb.	5	1	2
pride	Amb.	6	5	7
pupil	Amb.	3	3	4
rabbit	Unamb.	4	3	3
ram	Amb.	9	4	8
rare	Amb.	6	1	17
rate	Amb.	7	6	13
reflect	Amb.	7	2	3
refrain	Amb.	3	3	3
river	Unamb.	1	3	6
ruler	Amb.	2	1	2
sack	Amb.	13	6	7
safe	Amb.	7	1	7
sage	Amb.	5	6	16
sane	Unamb.	2	7	11

Word	Ambiguity (Rodd)	Senses (WordNet)	Meanings (BERT)	Senses (BERT)
scrap	Amb.	7	3	3
screen	Amb.	16	2	15
seal	Amb.	15	4	8
season	Amb.	6	5	5
second	Amb.	15	1	19
seek	Unamb.	6	1	8
sense	Amb.	9	4	5
sentence	Amb.	4	1	2
shed	Amb.	6	3	4
sign	Amb.	20	6	6
spade	Amb.	4	7	8
speaker	Amb.	3	5	5
spell	Amb.	10	5	5
stable	Amb.	7	2	3
staff	Amb.	8	2	2
stag	Amb.	5	4	4
stalk	Amb.	8	3	3
stamp	Amb.	18	2	4
staple	Amb.	7	3	3
static	Amb.	5	1	3
stern	Amb.	7	5	5
store	Amb.	6	1	2
strand	Amb.	9	5	5
straw	Amb.	7	7	7
swallow	Amb.	11	5	5
swear	Amb.	5	1	2
task	Unamb.	4	4	4
temple	Amb.	4	2	4
tend	Amb.	3	1	2
tense	Amb.	8	2	3
tent	Unamb.	3	1	3
term	Amb.	8	2	3
terror	Unamb.	4	5	6
thief	Unamb.	1	3	3
throat	Unamb.	4	1	2
throw	Unamb.	20	6	7
tiger	Unamb.	2	5	6
toast	Amb.	6	2	4
travel	Unamb.	9	3	3
trial	Amb.	6	4	4
trust	Amb.	12	6	8
uniform	Amb.	6	1	2
unite	Unamb.	6	6	8
urban	Unamb.	2	4	5
vent	Amb.	7	8	12
vote	Unamb.	10	1	16
warn	Unamb.	4	1	2
watch	Amb.	13	8	16
weapon	Unamb.	2	1	2
winter	Unamb.	2	1	4
yard	Amb.	9	7	13
lorry	Unamb.	2	3	3

Modeling the Ordering of English Adjectives using Collaborative Filtering

Sagar Indurkhya
Massachusetts Institute of Technology
32 Vassar St.
Cambridge, MA 02139
indurks@mit.edu

Abstract

We present a novel procedure for acquiring productive knowledge of restrictions on adjective ordering from counts of adjective-bigrams, which are readily obtainable from natural language corpora. The procedure uses a model-based Collaborative Filtering (CF) algorithm, and is the first computational model of adjective-ordering to do so. We consider two widely-used model-based CF algorithms, Singular Value Decomposition and Non-negative Matrix Factorization. We evaluated the procedure by first training the underlying CF model on subsets of the largest publicly available dataset of English adjective-bigrams, the Google Books NGram database, and then measuring the model's capacity to predict the ordering of (unseen) pairs of adjectives. Our results show that both CF models exhibit good performance on the task of predicting adjective ordering. Moreover, the CF models consistently outperform a baseline model that is grounded in a ranking of adjectives intended to align with a (linear) hierarchy of adjective-classes, suggesting that CF models make use adjective-ordering data that does not neatly fit into (proposed) hierarchies of adjective-classes.

1 Introduction

Linguists have long been observing, studying and characterizing restrictions on the ordering of adjectives – e.g. native English speakers will say *"the big red ball"* and not *"the red big ball"* (Rizzi and Cinque, 2016; Trotzke and Wittenberg, 2019). Typically, linguists have grouped adjectives into semantic classes over which an ordering is proposed - e.g. (Goyvaerts, 1968; Vendler, 1968; Dixon, 1982; Shaw and Hatzivassiloglou, 1999; Cinque, 1994) propose an ordering over the following classes of adjectives (where $A{>}B$ indicates A precedes B):

SIZE > SHAPE > AGE > COLOR > PROVENANCE

Moreover, linguists have noted recurring patterns of restrictions on adjective ordering that appear over a wide and diverse range of languages, and these patterns can be assembled together to form a cross-linguistic hierarchy of adjectives that informs the ordering of adjectives that (directly) modify a noun phrase (Sproat and Shih, 1991; LaPolla and Huang, 2004; Trainin and Shetreet, 2021).[1] Given that these *tacit* restrictions on adjective ordering appear consistently across languages, it is puzzling how, and the degree to which, learners acquire this knowledge of language from the primary linguistic data. This study addresses this puzzle by introducing a novel procedure for acquiring *productive* knowledge of restrictions on adjective ordering.

Our procedure, implemented as a working computer program, takes as input adjective-bigram statistics listed in the *Google Books NGram database*, and outputs a (learned) model of adjective-ordering preferences. Importantly, the procedure learns a model of adjective-ordering preferences that is *productive* in that it can predict that a speaker would prefer to say *"the big red ball"* and not *"the red big ball"* even if it has never seen the adjective bigrams *"big red"* and *"red big"*, so long as the input data includes other adjective bigrams that involve *"big"* and *"red"*. In this way, the model output by the procedure goes beyond the null-hypothesis of learners simply repeating back what they have heard (Bar-Sever et al., 2018). The procedure centers on a model-based Collaborative Filtering (CF) algorithm that maps adjectives-ordering data to a low-dimensional embedding space where latent relationships between adjectives are surfaced; our approach is informed by earlier work suggesting that CF can be employed to model constituent selection more broadly (Indurkhya, 2021). Our experiments show that model-based CF algorithms perform well on

[1] This led to the *cartographic enterprise*, which aims (in part) to provide a syntactic accounting, in the form of a detailed map, of the observed cross-linguistic hierarchy of adjective-classes (Scott, 2002; Laenzlinger, 2005; Cinque, 2010; Shlonsky, 2010; Cinque and Rizzi, 2012; Cinque, 2014).

the task of predicting the ordering relation (if there is one) between the two (input) adjectives. Moreover, the CF algorithms substantively outperform a baseline model that computes the relative difference in the ranking of the two (input) adjectives – this baseline model is informed by prior work that suggested that a ranking of adjectives that correlates with a (linear) hierarchy of adjective-classes can be used to determine restrictions on adjective ordering. Our experiments thus suggest that model-based CF algorithms leverage adjective-ordering data that does not neatly align with proposed hierarchies of adjective-classes.

The remainder of this study is organized as follows. We begin by discussing prior work that (computationally) modeled restrictions on adjective ordering (see §2) and review established methods for CF (see §3). Next, we walkthrough the construction of an Adjective Ordering matrix from adjective-bigram data drawn from the Google Books NGram database, and make note of cylical orderings over adjective that defy the organization of adjectives into a hierarchy (see §4). We then detail the computational experiments central to this study, and analyze the information used by the CF models we evaluated (see §5). Finally, we conclude by discussing the broader implications of our study, especially in light of the minimal assumptions our approach aims to make (see §6).

2 Prior Studies of Adjective Ordering

Previous (computational) models of adjective ordering can broadly be construed as falling into one of two categories that are distinguished by what they aim to explain. The first category of work includes empirically-grounded methods that aim to explain the distribution of adjective orderings observed in corpus data. The second category of work includes models that encode some proposed measure of adjectives and that aim to explain the proposed cross-linguistic hierarchy of adjectives (from which the ordering of adjectives can be deduced).[2] Note that this two-fold categorization of prior work is not a strict dichotomy - e.g. work aiming to explain how a cross-linguistic hierarchy is learned may involve an empirically grounded approach. Let us now examine these two categories of prior work in more detail.

[2]As (Svenonius, 2008) notes, each adjective-class in the hierarchy can be mapped to a functional head that may be incorporated into a determiner phrase.

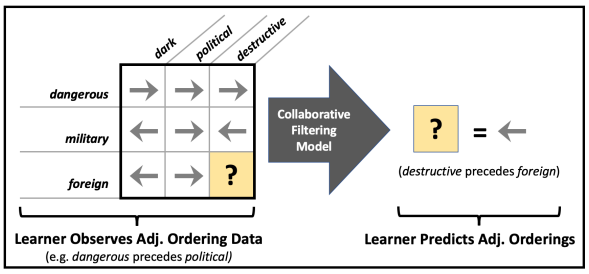


Figure 1: CF model for predicting the ordering of adjective pairs. Evidence supporting the prediction is: (i) *destructive* is similar to *dark* w.r.t. preceding *military* and following *dangerous*, and *dark* precedes *foreign*; (ii) *foreign* behaves like *military* w.r.t. preceding *political* and following *dark*, and *destructive* precedes *military*.

Prior work falling into the first category relies on corpus data from which the statistics of nouns and their adjectival pre-modifiers can be derived. Shaw and Hatzivassiloglou (1999) outline some of the standard methodologies employed, which we briefly describe here:

(a) *Direct Evidence* method: if the counts of adjective-bigrams "A B" and "B A" are c_{AB} and c_{BA} (respectively), a binomial test is used to determine whether the ratio of c_{AB} to c_{BA} differs significantly from the null-hypothesis of a 1:1 ratio – if the null-hypothesis is rejected, then we know speakers prefer "A B" over "B A" if $c_{AB} > c_{BA}$ (and vice versa). A weakness of this method is that it is *not* productive as it can only predict ordering preferences for adjective pairs that appeared as bigrams in the input corpus data.

(b) *Transitivity* method: given adjectives A, B and C, and evidence that $A > B$ and $B > C$ (perhaps determined via the binomial test outlined in the Direct Evidence method), this method will infer via transitivity that $A > C$. This method, which relates to the notion of ordering being derived from a hierarchy of adjective-classes, runs into difficulty when adjective ordering preferences inferred from the (input) corpus data violate transitivity. (See Table 1 for examples of adjective-bigram cycles.)

(c) *Clustering* method: adjectives are clustered together (e.g. via k-Nearest Neighbors) based on their ordering relation to other adjectives (with ordering determined via the binomial test as outlined in the Direct Evidence method), and the ordering of adjectives (A, B) is made by examining how other adjectives similar to A are ordered with respect to B. Similar to

memory-based CF algorithms (reviewed in §3), this approach suffers from sparsity of adjective-bigrams in the (input) corpus data, with many adjective pairings appearing only once in the data (Malouf, 2000).[3]

Although these empirically-grounded methods achieve reasonably good performance when applied to specific or restricted domains of text (where there are fewer occurrences of polysemy), their performance declines when applied to broader corpora of text that span multiple domains.

The second category of prior work aims to account for the (cross-linguistic) hierarchy of adjectives-classes. Some work in this category introduces a metric that can be used to rank adjectives, with the ordering over a pair of adjectives determined by comparing the rank of the two adjectives. For example, Hahn et al. (2018) introduces a metric that computes the mutual information between an adjective and the nominal head it modifies, so that given a nominal head, N, adjectives having higher mutual information with N will appear closer to N; this metric was grounded in the subjectivity theory of Hill (2012) and Scontras et al. (2017), with the latter carrying out experiments showing that *property-subjectivity* is what predicts adjective ordering - i.e. adjectives that are less subjective appear closer to the (modified) nominal head.[4] Alternatively, other work takes the approach of directly inferring a hierarchy of adjective classes. For example, recent work by Leung et al. (2020) demonstrates that latent-variable models (that predict adjective ordering) can learn cross-linguistic adjective hierarchies that align with Cinque's hierarchy – their model represents adjectives using *multi-lingual* word embeddings, which can (implicitly) encode information pertaining to adjective ordering in the form of the presence or absence of (nearby) non-adjectival tokens - e.g. the non-adjectival tokens that form the context of a word (that is being embedded) may encode the semantic category the word belongs to, in turn allowing it to be placed within Cinque's hierarchy.[5]

To summarize, this study primarily falls under the first category of work, although it is informed by prior work from both categories of prior work in so far as: (i) we employ a binomial test to determine adjective ordering from bigram counts as in Shaw and Hatzivassiloglou (1999); (ii) based on the assessment of Malouf (2000), we opt to directly work with the adjective-bigram counts listed in the *Google Books NGram database*, and thus do not consider the words surrounding the adjective-bigrams (see §4); (iii) the baseline model we test the CF models (output by our procedure) against is informed by observations in (Scontras et al., 2017; Hahn et al., 2018) that adjectives can be ranked, with an adjective's ranking determining it's ordering relative to other adjectives (and with the ranking meant to correlate with the hierarchy of adjective-classes). Finally, as we discuss in §3, our decision to use model-based CF algorithms is motivated in part by the need for a *productive* model that robustly deals with *sparse* (adjective-bigram) data drawn from a single language (i.e. no use of multilingual word embeddings), builds on the ideas underlying the *memory-based* models referenced in Shaw and Hatzivassiloglou (1999), and does not make assumptions about the *transitivity* of restrictions on adjective ordering.

3 A Review of Collaborative Filtering

The Collaborative Filtering (CF) algorithms used in this study are (widely used) examples of Recommender Systems (Herlocker et al., 2004; Su and Khoshgoftaar, 2009; Lü et al., 2012; Bobadilla et al., 2013). Given a finite set of users (e.g. subscribers), a finite set of items (e.g. movies), and a *user-item rating matrix* that encodes ratings assigned by (some) users to (some) items, a Recommender System is tasked with predicting the rating a given user would assign to a given item; these predictions may then be used to enumerate a list of recommended items for the given user. This study takes the users to be the first adjective in an adjective bigram, the items to be the second adjective in an adjective bigram, and the *user-item rating matrix* to be an *adjective-ordering matrix* (detailed

[3] Malouf (2000) also notes that trying to incorporate information from the text surrounding an adjective bigram doesn't provide sufficient syntagmatic information because the left side is usually a determiner and the right hand side (typically a nominal) only worsens the data sparsity issue.

[4] Note that this approach only partly aligns with there being a *hierarchy* of adjective classes as the experiments in Scontras et al. (2017) showed that users sometimes have differing subjectivity ratings for adjectives – this aligns with our observation (detailed in §4) that there are large numbers of adjective pairings for which there is no clearly preferred ordering. (See also Scontras et al. (2019).)

[5] The experiments detailed in Leung et al. (2020) control for explicitly encoded information about adjective ordering in the text from which the word-embedding models are learned; they do not, however control for implicitly encoded information in the form of the presence or absence of non-adjectival tokens incorporated into the word-embedding model.

Adj. Type	Examples		
	Ordered Adj. Pairs	Unordered Adj. Pairs	Adj. Cycles of Length 3
all	(important, private), (abbreviated, new)	(relative, domestic), (foreign, strategic)	(permanent, unconscious, young), (acoustic, grand, old)
pure	(intriguing, new), (international, technological)	(recent, substantial), (everyday, practical)	(ethical, foreign, institutional), (elementary, historical, prospective)
noun	(independent, moral), (radical, socialist)	(overall, specific), (fundamental, common)	(aged, former, intermediate), (medium, stiff, ordinary)
verb	(certified, registered), (corresponding, free)	(dry, round), (open, parallel)	(flat, major, long), (appropriate, major, free)

Table 1: Examples for each subset of the adjective-bigram data. An ordered pair of adjectives, (a, b), indicates that a typically precedes b (as determined by the test outlined in §4). An unordered pair of adjectives, (a, b), indicates that there is no preference of either a preceding b or b preceding a. Finally, an adjective cycle, (a, b, c), indicates the presence of three ordered pairs, $\{(a, b), (b, c), (c, a)\}$, that together form a cycle (and cannot be folded into a strictly linear hierarchy of adjectives).

in §4) – then a Recommender System may be used to predict, for a given adjective, which adjectives may follow it to form an adjective bigram.

Recommender Systems are typically divided into *Content-Based (CB) recommendation algorithms* and *CF recommendation algorithms*.[6] CB recommendation algorithms make use of similiarities between the valuations of features associated with each item (or alternatively, between the valuations of features associated with each user). For example, a CB recommendation algorithm can predict whether the adjective *"red"* can follow the adjective *"large"* by comparing the syntactic and semantic features associated with *"large"* with those associated with other adjectives known to follow *"large"* (such as *"blue"*). If these semantic and syntactic features are unavailable – e.g. as in this study, in which we intentionally restrict our attention to adjective bigram statistics derived from (unannotated) corpus data – then we can instead use CF recommendation algorithms, which can simultaneously take into account: (i) similarities between users, as measured by how similarly they rate items; (ii) similarities between items, as measured by how similarly they are rated by users. Conventionally, CF algorithms are grouped into two classes, *memory-based* CF algorithms and *model-based* CF algorithms, which we will now describe in turn.

Memory-based CF algorithms operate on the assumption that the more similar two users are (with respect to the ratings they assign), the more likely they are to assign similar ratings to items. A Memory-based CF algorithm predicts the rating a user, u, would assign to an item, i, by: (i) identifying a set of users, S, that are similar to u – e.g. by computing the *k Nearest Neighbors* using a similarity measure such as the Pearson correlation coefficient; (ii) computing the predicted rating as the weighted average of ratings assigned by each $s \in S$ to i, with the weights corresponding to the degree-of-similarity of each s to u (Schafer et al., 2007). More broadly, a memory-based CF algorithm is either an instance of *user-based* collaborative filtering, in which case it operates by identifying similar users (as described above), or alternatively, *item-based* collaborative filtering, in which case it identifies similar items. See (Sarwar et al., 2001) for a review of *item-based* CF.

Despite enjoying widespread usage, memory-based CF algorithms struggle in two scenarios: first, the quality of their predictions quickly degrades when the user-item rating matrix is sparse, and second, they do not scale well (with respect to memory consumption) as the number of users and items grow (Adomavicius and Tuzhilin, 2005). These difficulties are addressed by **Model-based CF algorithms**, which center on a *learned* predictive model. Notable examples of model-based CF algorithms include *Non-negative Matrix Factorization* (NMF) and *Singular Value Decomposition* (SVD). NMF and SVD, both instances of *latent factor models*, work by factoring apart the *user-item rating matrix*, so that the user and item profiles (corresponding to the rows and columns of the *user-item rating matrix*) are embedded in a lower-dimensional space where latent relationships

[6] (Burke, 2002, 2007) detail hybrid recommender systems that fuse together aspects of CB and CF algorithms.

between users and items are more readily apparent; in this way, NMF and SVD address two weaknesses of memory-based CF algorithms, *sparsity* and *scalability*. Finally, we note that NMF and SVD yield *linear* models that have the capacity to encode a hierarchy of adjective-classes (e.g. Cinque's hierarchy).[7] For these reasons, the present study opts to use two (latent factor) model-based CF algorithms, NMF and SVD, to model adjective-ordering.

4 Constructing an Adj. Ordering Matrix

We now detail how to derive an adjective-ordering matrix (i.e. the CF-model's input) using bigram data taken from the American-English (2019) subset of the *Google Books NGram database* (for the years 1969-2019) (Michel et al., 2011; Lin et al., 2012).

To begin, we define a *word pair* to be a two-tuple of words, (x, y), such that x lexicographically precedes y. We restricted our analysis to only consider a word pair (x, y) if met three conditions:

(a) let A be the set of all adjectives appearing in the Google Books NGram database (as marked by the Part-of-Speech tagging of each ngram in the database), then $x, y \in A$;

(b) the sum of the frequencies of the bigrams "x y" and "y x" is at least 100 – this eliminates word pairs with insufficient samples for statistical hypothesis testing;

(c) let W be the set of adjectives in the WordNet database (Miller et al., 1990; Miller, 1995), and let W' be the set of adjective lemmas obtained by lemmatizing members of W – then $lemma(x) \in W'$ and $lemma(y) \in W'$.[8][9]

There were 491499 distinct word pairs that met these conditions.

Next we define an *adjective pair* as a two-tuple of lemmas, (s, t), such that s and t are both lemmas of adjectives that make up a word pair; there are 472823 distinct adjective pairs, and 10041 distinct adjective-lemmas appeared in these adjective pairs. Given an adjective pair (s, t), we define its *forward frequency*, $f(s, t)$, as the sum of the frequencies of bigrams of the form "x y" where $s = lemma(x)$ and $t = lemma(y)$.[10] Likewise, we define the *backward frequency*, $b(s, t)$, as the sum of the frequencies of bigrams of the form "y x" where $s = lemma(x)$ and $t = lemma(y)$. Finally, we define the *ratio*, $r(s, t)$, as $\frac{f(s,t)}{f(s,t)+b(s,t)}$.

We then marked each adjective pair as either being *ordered* or *unordered*. To make this determination for an adjective pair (s, t), we used a two-tailed binomial test, with probability of success $r(s, t)$, to evaluate the null hypothesis that bigrams "x y" and "y x", where $s = lemma(x)$ and $y = lemma(y)$, are equally likely to appear (because there is no significant preference in the ordering of the two adjectives, and thus the adjective pair is determined to be *unordered*). We rejected the null hypothesis (implying the adjective pair is *ordered*) if $p < \frac{0.01}{n^2}$, where $n = 10041$ is the number of distinct (lemmatized) adjectives and n^2 is the Bonferroni correction factor.[11] Of the adjective pairs, 344228 were found to be ordered and 128595 were found to be unordered. Table 1 shows examples of ordered and unordered adjective-pairs.[12]

We can now define the *adjective-ordering matrix*, Q (a 10041×10041 matrix), as follows. Let L be the set of adjectives pairs. Then for each $(s, t) \in L$, (i) if (s, t) is unordered, then $Q[s, t] = Q[t, s] = 1$; (ii) if (s, t) is ordered, then $Q[s, t] = 2$ if $r(s, t) > 0.5$, otherwise $Q[t, s] = 2$. Any entry in Q not defined above has value 0.[13]

To better understand how the (alternative) lexical categories that a polysemous adjective can appear as may impact a model's ability to learn ordering information,[14] we also produced adjective-ordering matrices for subsets of the set of adjective pairs - these subsets are labeled as follows: *all* denotes

[7] This is possible if each adjective is mapped (via it's embedding vector) to a weighted sum of adjective classes – then the learned matrix can encode information about which adjective classes can precede which other adjective classes.

[8] Checking for membership in WordNet's list of adjectives guards against potential mislabeling of Part-of-Speech tags in the Google Books NGram database, spelling errors, etc, and enables identification of adjectives that are also verbs or nouns (using WordNet's lists of nouns and verbs).

[9] The lemmatized form serves to normalize the various forms of adjectives (e.g. superlatives and comparatives). of W'. We used spaCy (v3.2.0) to lemmatize words.

[10] The frequency of a bigram is the number of times it appears in the Google Books corpus, as recorded in the Google Books Ngram database. Our measurement of bigrams frequency was case-insensitive.

[11] Bonferroni correction counteracts the propensity for false-positives arising when doing multiple comparison testing.

[12] The reader can inspect an example of an adjective pair (a, b) by going to the *google-ngrams* website (`https://books.google.com/ngrams`) and running the query "`a_ADJ b_ADJ, b_ADJ a_ADJ`" with the queried data restricted to the years 1969-2019.

[13] We use values of 2 and 1 so that the adjective-ordering matrix can serve as input to the Non-negative Matrix Factorization (NMF) CF model; the CF models ignore the 0 values.

[14] E.g. the polysemous word *"sound"* can appear as a noun or as an adjective, each having a different meaning. See also (Taylor, 2003; Baker and Baker, 2003; Falkum, 2015).

Adj. Type	Adjective-Ordering Matrix Statistics				Model Performance (AUROC)			t-Test Statistic	
	Adjs.	Adj. Pairs	Ordered Pairs	Cycles	NMF	SVD	Baseline (γ)	NMF	SVD
all	10041	472823	72.8%	970527	**0.788**	0.770	0.716	28.972	33.420
pure	6817	76228	70.3%	15661	**0.756**	0.738	0.676	5.915	26.799
noun	2688	130701	75.6%	311491	**0.767**	0.734	0.710	23.431	10.143
verb	1755	26284	71.2%	22430	**0.716**	0.689	0.662	10.426	13.654
acyclic-all	9296	382335	66.6%	0	**0.867**	0.862	0.791	45.505	46.278
acyclic-pure	4776	64843	66.3%	0	**0.813**	0.809	0.755	6.239	27.481
acyclic-noun	2450	105938	70.0%	0	**0.858**	0.853	0.795	29.530	9.344
acyclic-verb	1265	21657	65.8%	0	**0.797**	0.788	0.758	10.367	9.586

Table 2: Performance of two Collaborative Filtering models (NMF and SVD) and a baseline (γ) model on each adjective-ordering matrix that was derived from the (Google NGrams) Adjective Bigram data. Notably, the NMF model consistently has the top performance, and both Collaborative Filtering models consistently outperform the baseline model. *Adj. Type* indicates the subset of the data from which an adjective-ordering matrix was derived and whether or not adjective cycles are present. Key statistics are presented for each adjective-ordering matrices, and for each (model, dataset) pair, we report for the median-scoring model: (i) the performance (as measured by AUROC), and (ii) the Welch's t-test statistic that is used to compare the distributions of the *fraction of adjacent adjective pairs* for correctly and incorrectly classified adjective pairs (see the analysis in §5 for details).

the original set of adjective pairs; *noun* denotes the subset of *all* in which both lemmas in an adjective pair appear in the lemmatized list of nouns in WordNet; likewise, *verb* denotes the subset of *all* in which both lemmas in an adjective pair appear in the lemmatized list of verbs in WordNet; *pure* denotes the subset of *all* in which both lemmas in an adjective pair are not members of the *noun* or *verb* subsets. Table 2 presents statistics for each adjective-ordering matrix.

Finally, we note that the *ordered* adjective pairs can be modeled as a directed graph: an adjacency matrix encoding a directed graph, where nodes correspond to distinct lemmatized adjectives, may be obtained by removing the entries in Q associated with unordered adjective pairs. Upon constructing these graphs, we identified the presence of *adjective cycles*, which are cycles of edges in the graph formed by a sequence of adjective bigrams - see Table 1 for examples of these cycles, and see Table 2 for counts of adjective cycles of length three. *The presence of these adjective cycles surprised us as it seems to rebut the ordering hierarchy over adjectives proposed by the Cartographic Enterprise.* To better understand the role that the presence of these cycles may play in a model's ability to learn the ordering of adjectives, we constructed a directed acyclic graph (DAG) that was a subgraph of the directed graph derived from the adjective-ordering matrix. We did this by identifying a *feedback arc set*, which is a set of adjective pairs that contains at least one adjective pair in each cycle in the graph, using the method introduced by (Eades et al., 1993); note that this method divides the original directed graph into two DAGs, and we selected the larger DAG. The resulting DAG was then used to reformulate an adjective-ordering matrix, with the entry for an *unordered* adjective pair (s, t) carried over from the source adjective-ordering matrix if s and t were both present in the DAG. This process was repeated for each subset of adjective pairs, yielding four new subsets labeled *acyclic-noun*, *acyclic-verb*, *acyclic-all* and *acyclic-pure* (see Table 2 for details).

5 Experiment

We evaluated two different latent factor model-based Collaborative Filtering methods (Hofmann, 2004; Koren et al., 2009), *Singular Value Decomposition* (SVD) and *Non-negative Matrix Factorization* (NMF), on the task of predicting, given adjective pair data in the training set, whether a given adjective pair (in the test set) is ordered, and if it is ordered, which direction the ordering is in.[15]

Methodology. Given an adjective-ordering matrix, we trained each of the two CF models by employing nested 5-fold cross-validation with shuffling, in which the outer loop evaluates trained models, and the inner loop is used for model selection (hyperparameter tuning) and model fitting (i.e. training). Specifically, the outer-loop consists of 5-fold cross-validation, with 20% of the data (i.e. entries in the adjective-ordering matrix) held out as a test

[15]We used the implementations of NMF and SVD provided in the (python) library *Surprise* (v1.1.1) (Hug, 2020). See Appendix A for details about the computing infrastructure, software libraries and runtime used for these experiments.

dataset and the remaining data used for training and validation; the inner-loop consists of 5-fold cross-validation with 80% of the data used as a training set and the other 20% of the data held out as a validation set.[16] We evaluated performance during model selection by computing the mean average error (MAE), a metric that is commonly used for evaluating model-based CF algorithms.[17]

We trained and evaluated each of the two CF-models on each of the eight adjective-ordering matrices listed in Table 2. Note that a trained CF model, M, consists of: (i) a mapping, u_M, between (lemmatized) adjectives and embedding vectors of length n_f+1 – this mapping encodes the *first* item in an adjective pair (i.e. the s in an adjective pair (s,t)) into an embedding vector; (ii) a second mapping, v_M, between (lemmatized) adjectives and embedding vectors of length n_f+1 – this mapping encodes the *second* item in an adjective pair into an embedding vector; (iii) an $(n_f+1)\times(n_f+1)$ matrix, S_M – in the case of NMF, S_M is the identity matrix. The model estimates the value for an adjective bigram (from the test set), (s,t), as:

$$M(s,t) = u_M(s)S(v_M(t))^T$$

Given two adjectives a and b, the procedure for determining adjective ordering (which encapsulates the CF-model M) makes a prediction, $P_{\{a,b\}}$, about the ordering relationship between a and b as:

$$P_{\{a,b\}} = \begin{cases} A > B, & \text{if } M(a,b) \geq \psi > M(b,a) \\ A < B, & \text{if } M(a,b) < \psi \leq M(b,a) \\ \text{No Ordering}, & \text{Otherwise} \end{cases}$$

Here ψ is a threshold with $1 \leq \psi \leq 2$, such that we classify $M(s,t)$ as a *high* value (2) if $M(s,t) \geq \psi$ and a *low* value (1) otherwise. As the accuracy of the model depends on the value ψ, we thus evaluated model-performance by computing the Area Under the Receiver Operating Characteristic (AUROC) curve (Fawcett, 2006).

Results. Table 2 summarizes the results of our experiments. Notably, the CF models, NMF and SVD, achieved high AUROCs of 0.87 and 0.86 (respectively) on the *acyclic-all* adjective-pair data, and 0.79 and 0.78 (respectively) on the *all* adjective-pair data. Overall, NMF achieved the highest AUROC on each of the adjective pairs datasets. We also observed that both NMF and SVD performed better on the *acyclic* datasets than on their cyclic counterparts, and that for datasets both with and without cycles, both CF-models performed better on larger and less restricted datasets, *all* and *noun* (c.f. the smaller, more restricted datasets, *pure*).

We also evaluated a baseline model, referred to as the γ baseline, that serves as a reference point against which we compared the performance of the CF models. The γ baseline, which takes into account both input adjectives, is defined as follows. For an adjective z, let $\rho_{z,1}$ be the multiset of values for entries in the training data[18] where the first adjective is z, let $\rho_{z,2}$ be the multiset of values for entries in the training data where the second adjective is z, and let $h(z)$ be the weighted harmonic mean of $avg(\rho_{z,1})$ and $(3 - avg(\rho_{z,2}))$, so that:

$$h(z) = \frac{|\rho_{z,1}| + |\rho_{z,2}|}{|\rho_{z,1}|(\overline{\rho_{z,1}})^{-1} + |\rho_{z,2}|\,(3 - (\overline{\rho_{z,2}}))^{-1}}$$

Here, $h(z)$ is a ranking of the adjective z that is intended to correlate with z's position in the hierarchy of adjective-classes.[19] Given adjective pair (s,t) in the test data, the γ baseline is a linear transform[20] of the difference between the rankings of the two adjectives s and t:

$$\gamma(s,t) = \frac{3}{2} + \frac{1}{2}(h(s) - h(t))$$

Importantly, by using the γ baseline (in place of the CF models), our procedure can predict the ordering of adjectives, s and t, by comparing their rankings (per h) – importantly, this grounds the γ baseline in the second category of work described in §2.

Notably, the two CF models, NMF and SVD, outperformed the γ-baseline on each of the eight adjective-ordering matrices (see Table 2). We also observed that the γ baseline performed better on the *acyclic* datasets than on the full datasets – this was not surprising as the γ baseline is a model of a linear hierarchy of adjectives (i.e. $h(z)$ forms a *total preordering* over adjectives). Overall, our results show that both model-based CF algorithms: (i) perform well on the task of predicting adjective ordering, and (ii) outperform a baseline model

[16] An adjective-ordering matrix Q can be represented as a set of tuples of the form $(A, B, Q_{A,B})$ where A is an adjective coding for a row, $r_A \in Q$, B is an adjective coding for a column, $c_B \in Q$, and $Q_{A,B}$ is the value $Q[r_A, c_B]$ – then (A, B) is the model's input, and $Q_{A,B}$ is the model's output.

[17] Model selection for both models, NMF and SVD, involved optimizing the hyperparameter for the number of latent factors, $n_f \in \{4, 6, 8, \ldots, 16, 18\}$, and both models were trained for 450 epochs. The NMF model used a regularization rate of 0.06, and the SVD model used a learning rate of 0.005 and a regularization rate of 0.02.

[18] The training data does not include any zero-valued entries in the adjective-ordering matrix from which it is derived.

[19] N.b. the greater the number of adjectives that z precedes (as measured by $\overline{\rho_{z,1}}$), and the fewer the number of adjectives that precede z (as measured by $3-\overline{\rho_{z,2}}$), the larger $h(z)$ is.

[20] As $h(z) \in [1, 2]$, this transform ensures $\gamma(s,t) \in [1, 2]$.

grounded in a ranking (of adjectives) meant to correlate with a (linear) hierarchy of adjective-classes.

Analysis. We analyzed the degree to which the CF models, when predicting adjective ordering, utilize information about related adjective pairs. Let ω be the set of adjective pairs (in the training data), and let ψ be the set of all adjectives appearing in ω. Given an adjective pair (s, t), we define its *adjacent adjective pairs* (AAP) as the set $\mu_{(s,t)} \cap \omega$ where:

$$\mu_{(s,t)} = \{(x, y) \in \psi \times \psi | (s, y) \in \omega \wedge (x, t) \in \omega\}$$

We define the *fraction of adjacent adjective pairs* (FAAP) for (s, t) as the ratio:

$$|\mu_{(s,t)} \cap \omega| / |\mu_{(s,t)}|$$

Given an adjective pair (s, t), FAAP is the ratio of AAP present in the training data vs. the maximum number of AAP that could have been in the training data; the smaller the FAAP of (s, t) is, the more discriminating s and t are in the adjectives they appear with (in a bigram).

We now consider, for each (model, dataset) pair, how FAAP relates to model performance. We first computed the threshold[21] that maximized the model's F1-score, and then determined, for each adjective pair (s, t) in the test data, whether the model's output, $M(s, t)$, was correct (as classified by $P_{s,t}$).[22] We then computed the means and variances for the distributions of FAAP for adjective pairs that were correctly and incorrectly (respectively) classified, and we found that the mean of the former distribution was *consistently lower* than the mean of the latter.[23] We thus evaluated whether these two distributions of FAAP differed substantively. We used Welch's t-test (Welch, 1947) to test the null-hypothesis (α=0.01) that there is no (statistically) significant difference between the means of the two distributions (e.g. see Fig. 2); in each case, the null-hypothesis was rejected as the p-value was at most 3.5×10^{-9}, which was well below the critical value (α). We inferred that, consistently, the means of the two distributions differ significantly.

Moreoever, the t-test statistic appeared to correlate with the model's maximum F1-score, and is generally greater in the acyclic datasets (cf. the cyclic datasets) and in the *all* and *noun* subsets (cf. *verb* and *pure*). To validate this observation, we used linear regression to analyze the correlation between the t-test statistic of a model and the optimal F1-score of the model, which yielded a coefficient of determination (R^2) of 0.61 (p=3.7×10^{-4}). This suggests that the difference between the means of the correct and incorrect FAAP distributions is a significant factor in explaining model performance.

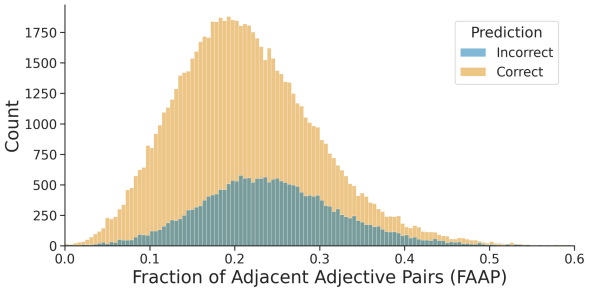


Figure 2: Given the ordering classifications made by the NMF model on the *all* test set, the distributions of FAAP for *correctly* and *incorrectly* classified adjective pairs have (mean, variance) of (0.237, 0.008) and (0.253, 0.008) respectively. Welch's t-test yields a test statistic of 28.97 and a p-value of 1.93×10^{-183}, which falls well below a critical p-value of 0.01 – hence, the means of the two distributions differ significantly.

6 Conclusion

This study showed that CF models, when trained on adjective bigram data readibly obtained from a corpus, perform well on the task of predicting the ordering of (unseen) adjective pairs. We compared the CF models with a baseline model, γ, that is grounded in a ranking of adjectives intended to parallel a (linear) hierarchy of adjective-classes. Notably, our results show that the CF models (consistently) perform markedly better than the baseline model, suggesting that CF models leverage adjective-ordering data that does not neatly align with proposed hierarchies of adjective-classes.

More broadly, the present study was motivated by the desire to see how far *productive* corpus-based models can be taken when they: (i) are restricted to adjective bigram statistics, (ii) do not require retention of the entire adjective-cooccurrence matrix after training, (iii) are robust in the face of sparse datasets, and (iv) must make predictions about the ordering of previously unseen adjective pairs. Moreoever, we made minimal assumptions about a learner's innate knowledge of language, the kind of data they have access to, and the size of their memory. To this extent, our procedure is a baseline that other models should aim to surpass to better justify any stronger assumptions they make.

[21] I.e. the threshold used to classify the model's (continuous) output as 1 (low) or 2 (high).

[22] We used, for each (model, dataset) pair, the model instance with median AUROC during cross-validation.

[23] Table 3 in the appendix details these distributions and lists the maximum F1-score for each (model, dataset) pair.

References

Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749.

Artemis Alexiadou. 2001. Adjective syntax and noun raising: Word order asymmetries in the dp as the result of adjective distribution. *Studia linguistica*, 55(3):217–248.

Mark Baker and William Croft. 2017. Lexical categories: Legacy, lacuna, and opportunity for functionalists and formalists. *Annual Review of Linguistics*, 3:179–197.

Mark C Baker and Mark Cleland Baker. 2003. *Lexical categories: Verbs, nouns and adjectives*, volume 102. Cambridge University Press.

Galia Bar-Sever, Rachael Lee, Gregory Scontras, and Lisa Pearl. 2018. Little lexical learners: Quantitatively assessing the development of adjective ordering preferences. In *Proceedings of the 42nd annual Boston university conference on language development*, pages 58–71. Cascadilla Press Somerville, MA.

Judy B Bernstein. 1993. *Topics in the syntax of nominal structure across Romance*. City University of New York.

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems*, 46:109–132.

Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370.

Robin Burke. 2007. Hybrid web recommender systems. *The adaptive web*, pages 377–408.

Guglielmo Cinque. 1994. On the evidence for partial n-movement in the romance dp. In *Paths towards universal grammar. Studies in honor of Richard S. Kayne*. Georgetown University Press.

Guglielmo Cinque. 2010. *The syntax of adjectives: A comparative study*, volume 57. MIT press.

Guglielmo Cinque. 2014. The semantic classification of adjectives. a view from syntax. *Studies in Chinese Linguistics, 35.(1) 3-32*.

Guglielmo Cinque and Luigi Rizzi. 2012. The cartography of syntactic structures. In *The Oxford handbook of linguistic analysis*, pages 51–65. Oxford University Press.

Robert M. W. Dixon. 1982. Where have all the adjectives gone? *Berlin: Mouton de Gruyter*.

Peter Eades, Xuemin Lin, and William F Smyth. 1993. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47(6):319–323.

Ingrid Lossius Falkum. 2015. The how and why of polysemy: A pragmatic account. *Lingua*, 157:83–99.

Tom Fawcett. 2006. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

Didier L Goyvaerts. 1968. An introductory study on the ordering of a string of adjectives in present-day english. *Philologica Pragensia*, 11(1):12–28.

Michael Hahn, Judith Degen, Noah D Goodman, Dan Jurafsky, and Richard Futrell. 2018. An information-theoretic explanation of adjective ordering preferences. In *CogSci*.

Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53.

Felix Hill. 2012. Beauty before age? applying subjectivity to automatic english adjective ordering. In *Proceedings of the NAACL HLT 2012 student research workshop*, pages 11–16.

Thomas Hofmann. 2004. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115.

Nicolas Hug. 2020. Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174.

Sagar Indurkhya. 2020. Inferring minimalist grammars with an SMT-solver. *Proceedings of the Society for Computation in Linguistics*, 3(1):476–479.

Sagar Indurkhya. 2021. Using collaborative filtering to model argument selection. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 629–639.

Sagar Indurkhya. 2022. Incremental acquisition of a Minimalist Grammar using an SMT-solver. In *Proceedings of the Society for Computation in Linguistics 2022*, pages 212–216, online. Association for Computational Linguistics.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Christopher Laenzlinger. 2005. French adjective ordering: Perspectives on dp-internal movement types. *Lingua*, 115(5):645–689.

Randy J LaPolla and Chenglong Huang. 2004. Adjectives in qiang. *Adjective classes: A cross-linguistic typology*, pages 306–322.

Jun Yen Leung, Guy Emerson, and Ryan Cotterell. 2020. Investigating Cross-Linguistic Adjective Ordering Tendencies with a Latent-Variable Model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4016–4028, Online. Association for Computational Linguistics.

Yuri Lin, Jean-Baptiste Michel, Erez Aiden Lieberman, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the google books ngram corpus. In *Proceedings of the ACL 2012 system demonstrations*, pages 169–174.

Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. 2012. Recommender systems. *Physics Reports*, 519(1):1–49. Recommender Systems.

Brian MacWhinney. 2000. *The CHILDES project: The database*, volume 2. Psychology Press.

Brian MacWhinney and Catherine Snow. 1985. The child language data exchange system. *Journal of child language*, 12(2):271–295.

Robert Malouf. 2000. The order of prenominal adjectives in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 85–92.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Google Books Team, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, et al. 2011. Quantitative analysis of culture using millions of digitized books. *science*, 331(6014):176–182.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.

Luigi Rizzi and Guglielmo Cinque. 2016. Functional categories and syntactic theory. *Annual Review of Linguistics*, 2:139–163.

Alessandro Sanchez, Stephan C Meylan, Mika Braginsky, Kyle E MacDonald, Daniel Yurovsky, and Michael C Frank. 2019. childes-db: A flexible and reproducible interface to the child language data exchange system. *Behavior research methods*, 51(4):1928–1941.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer.

Gregory Scontras, Judith Degen, and Noah D Goodman. 2017. Subjectivity predicts adjective ordering preferences. *Open Mind*, 1(1):53–66.

Gregory Scontras, Judith Degen, and Noah D Goodman. 2019. On the grammatical source of adjective ordering preferences. *Semantics and Pragmatics*, 12:7.

Gary-John Scott. 2002. Stacked adjectival modification. *Functional Structure in DP and IP: The Cartography of Syntactic Structures, vol*, 1:91–122.

James Shaw and Vasileios Hatzivassiloglou. 1999. Ordering among premodifiers. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 135–143.

Ur Shlonsky. 2010. The cartographic enterprise in syntax. *Language and linguistics compass*, 4(6):417–429.

Richard Sproat and Chilin Shih. 1991. The cross-linguistic distribution of adjective ordering restrictions. In *Interdisciplinary approaches to language*, pages 565–593. Springer.

Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009.

Peter Svenonius. 1994. C-selection as feature-checking. *Studia linguistica*, 48(2):133–155.

Peter Svenonius. 2008. The position of adjectives and other phrasal modifiers in the decomposition of dp. In *Adjectives and Adverbs: Syntax, Semantics, and Discourse*.

John R Taylor. 2003. Polysemy's paradoxes. *Language sciences*, 25(6):637–655.

Nitzan Trainin and Einat Shetreet. 2021. It's a dotted blue big star: on adjective ordering in a post-nominal language. *Language, Cognition and Neuroscience*, 36(3):320–341.

Andreas Trotzke and Eva Wittenberg. 2019. Longstanding issues in adjective order and corpus evidence for a multifactorial approach. *Linguistics*, 57(2):273–282.

Zeno Vendler. 1968. Adjectives and nominalizations.

Bernard L Welch. 1947. The generalization of Student's 'problem when several different population varlances are involved. *Biometrika*, 34(1-2):28–35.

Charles Yang. 2011. Computational models of language acquisition. In *Handbook of generative approaches to language acquisition*, pages 119–154. Springer.

A Computing Infrastructure and Runtime

The experiments described in this paper were carried out on a linux server with the following specifications: Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz; 54 GB of RAM; 1 TB of HDD. We used Python (v3.9.7) and the following libraries: *pandas* (v1.2.3) and *matplotlib* (v3.4.3), and *scipy* (v1.6.2); we used the implementation of Student's t-test and linear regression found in the python library, *scipy* (v1.6.2). The data-processing and experiment (i.e. model selection and training, evaluation and analysis-routines) took $\approx$30 hours of total runtime.

B Applications

The procedure introduced in this study makes minimal assumptions about a learner's innate knowledge of language, the kind of data they have access to, and the size of their memory. For this reason, we believe that upon further analysis of the procedure's performance on smaller corpora that reflect the primary linguistic data a child learner would encounter (MacWhinney and Snow, 1985; MacWhinney, 2000; Sanchez et al., 2019), our procedure may prove to be a suitable candidate for augmenting computational models of child language acquisition – see (Yang, 2011) for a review of computational models of language acquisition, and see (Indurkhya, 2020, 2022) for examples of models that this procedure could augment. Specifically, the procedure may be used to augment models that aim to explain how children acquire knowledge of restrictions on *constituent selection* (Svenonius, 1994) when forming syntactic structures, and in particular, when forming the *fine structure* of the Determiner Phrase (DP) in which a sequence of adjectives may be (hierarchically) embedded (Bernstein, 1993; Alexiadou, 2001; Baker and Croft, 2017). E.g. the productive model learned by this procedure can be used to score (or rank) candidate sequences of adjectival modifiers within a syntactic structure produced via a generative procedure, thereby serving as a filter on the production of adjective bigrams that the learner did not see in the primary linguistic data.

Model	Adj. Type	AUROC	Optimal F1-score	Optimal Threshold	FAAP for Correct Classification			FAAP for Incorrect Classification			t-test statistic	p-value
					Mean	Variance	Support	Mean	Variance	Support		
NMF	acyclic-all	0.867	0.794	1.345	0.215	0.007	78524	0.243	0.007	22978	45.505	0.000e+00
NMF	acyclic-pure	0.813	0.751	1.310	0.156	0.009	12433	0.167	0.010	4544	6.239	4.651e-10
NMF	acyclic-noun	0.858	0.809	1.365	0.280	0.007	21247	0.315	0.007	6171	29.530	7.752e-184
NMF	acyclic-verb	0.797	0.747	1.310	0.259	0.010	4156	0.289	0.009	1580	10.367	9.368e-25
NMF	all	0.788	0.764	1.375	0.237	0.008	84015	0.253	0.008	35554	28.972	1.926e-183
NMF	pure	0.756	0.736	1.280	0.165	0.009	12944	0.174	0.009	6355	5.915	3.414e-09
NMF	noun	0.767	0.773	1.370	0.297	0.007	22335	0.320	0.006	10026	23.431	8.223e-120
NMF	verb	0.716	0.726	1.210	0.285	0.011	4146	0.311	0.009	2514	10.426	3.208e-25
SVD	acyclic-all	0.862	0.794	1.395	0.214	0.007	78394	0.244	0.007	23113	46.278	0.000e+00
SVD	acyclic-noun	0.853	0.808	1.395	0.280	0.007	21132	0.312	0.007	6293	27.481	1.765e-160
SVD	acyclic-pure	0.809	0.748	1.390	0.153	0.008	12370	0.169	0.010	4567	9.344	1.201e-20
SVD	acyclic-verb	0.788	0.732	1.360	0.262	0.010	4047	0.289	0.009	1697	9.586	1.741e-21
SVD	all	0.770	0.757	1.370	0.235	0.008	81610	0.254	0.008	37983	33.420	4.391e-243
SVD	noun	0.734	0.766	1.300	0.296	0.007	21416	0.322	0.007	10924	26.799	9.675e-156
SVD	pure	0.738	0.727	1.270	0.163	0.009	12445	0.178	0.010	6806	10.143	4.372e-24
SVD	verb	0.689	0.724	1.155	0.278	0.010	4024	0.312	0.009	2644	13.654	8.414e-42

Table 3: A summary of the analysis of experiment results. Note that, for a given (model, dataset) pairing, the Optimal F1-score for the model was obtained by setting the model's classification threshold to the value listed under the Optimal Threshold. See §5 for additional details.

Comparison of Token- and Character-Level Approaches to Restoration of Spaces, Punctuation, and Capitalization in Various Languages

Laurence Dyer[1], Anthony Hughes[1], Dhwani Shah, Burcu Can

Research Group in Computational Linguistics (RCGL),
University of Wolverhampton, Wolverhampton, UK
{l.j.dyer, a.j.hughes2, d.r.shah2, b.can}@wlv.ac.uk

Abstract

This study compares token- and character-level approaches to restoration of spaces, punctuation, and capitalization to unformatted sequences of input characters, which is a vital step in use cases such as formatting outputs of automatic speech recognition systems for automatic transcription, and formatting hashtags. We obtain an overall F-score of 0.95 for our main English dataset with a hybrid pipeline model composed of a Naive Bayes classifier for space restoration and a BiLSTM classifier with a pre-trained Transformer layer for restoration of punctuation and capitalization. We also propose a novel character-level end-to-end BiLSTM model (overall F-score 0.90) which has the advantages of being able to restore mid-token capitalization and punctuation and of not requiring space characters to be present in input strings. We demonstrate that this model is language agnostic through experiments on Japanese, a *scriptio continua* language, and Gujarati, a low-resource language. We also compare our models with the only existing work on this task of which we are aware by carrying out experiments on the same dataset, and find that all of our models outperform those in that work.

1 Introduction

The accuracy of automatic speech recognition (ASR) systems has improved dramatically in recent years, with increasingly sophisticated deep learning architectures bringing about year-on-year improvements in word error rates (WERs) on established benchmark datasets such as LibriSpeech (Synnaeve, 2022). It has been claimed that ASR systems have reached the level of human parity (Xiong et al., 2016) or even super-human performance (Nguyen et al., 2020). The result has been a resurgence of various speech technologies, not least automatic transcription. Automatic transcription is widely used in the medical field (Van der Straten, 2017) and is now a hot topic in the general business community after many businesses shifted to fully remote or hybrid working following the COVID-19 pandemic. Transcription services integrated into online conferencing applications like Microsoft Teams and Zoom are used on a daily basis by organizations around the world to facilitate information sharing and record keeping.

However, most modern end-to-end ASR systems output streams of lowercase words without punctuation or capitalization (Guan, 2020), so these features must be restored by dedicated models in the post-processing stage. The presence or absence of certain punctuation marks can dramatically alter the understood meaning of a sentence—as anyone who has read the book *Eats, Shoots & Leaves* (Truss, 2004) knows—and the same set of letters can carry completely different meanings depending on the capitalization ("The Who" vs. "the WHO"). Capitalization and punctuation are also known to improve the readability of transcriptions (Jones et al., 2003). Moreover, absence of capitalization and punctuation negatively impacts the performance of downstream natural language processing (NLP) applications such as neural machine translation (Peitz et al., 2011), named entity recognition, and part-of-speech tagging (Mayhew et al., 2019). Appropriate punctuation and capitalization styles may vary depending on domain, so the ability to easily train models to restore these features is likely to become increasingly important in the future.

Most work on punctuation and capitalization restoration to date has been targeted at the token level, assuming that inputs are already correctly segmented into words. However, there are some

[1]Laurence Dyer and Anthony Hughes contributed equally to this work as first authors.

Repositories containing the code used for all of the experiments in this paper, together with comprehensive documentation and links to interactive demo notebooks, are available online at: https://ljdyer.github.io/Space-Punct-Cap-Restoration.

drawbacks to this approach. Firstly, it necessarily restricts the possible positions of each feature within a single word, and therefore cannot learn or restore mid-token punctuation or capitalization such as that found in "iPhone" or "yahoo.com". Secondly, it cannot be applied directly to *scriptio continua* languages like Chinese, Japanese, and Thai. Space restoration may also be required for certain use cases in spaced languages, for example when dealing with hashtags (Abd-hood and Omar, 2021).

For these reasons, in this study we assume raw data that comprises streams of lowercase characters and digits only, and examine a range of approaches to restoring spaces, punctuation, and capitalization to generate readable output texts. We compare pipeline approaches that first restore spaces and then tackle capitalization and punctuation at the token level with end-to-end approaches that perform all tasks simultaneously at the character level. We implement a combination of statistical and deep learning models in order to ascertain whether using neural network architectures leads to better performance in each part of the task under consideration.

2 Related Work

2.1 Space Restoration/Word Segmentation

Space restoration for spaced languages is largely equivalent to the problem of word segmentation for *scriptio continua* languages such as Chinese, for which a wealth of literature exists due to its applications in higher-level NLP tasks including part-of-speech tagging and machine translation. The problem can be tackled using dictionary-based approaches and statistical machine learning methods including Naive Bayes, support vector machines (SVM), and conditional random fields (CRF) (Haruechaiyasak et al., 2008). However, most recent work in this area employs neural networks.

Shao et al. (2018) propose a bidirectional recurrent neural network (RNN) architecture with gated recurrent units (GRU) that works at character level, and demonstrate its effectiveness on over 75 languages from a diverse range of language families. Sun (2010) compares word- and character-based approaches to Chinese word segmentation and proposes a classifier ensemble method after observing that each method leads to different types of error. Zhang et al. (2018) and Liu et al. (2019a) have proposed methods for incorporating information from dictionaries to improve neural network models. Higashiyama et al. (2019) integrate both word character information and character attention into a character-based neural network for Japanese language word segmentation. Zheng and Zheng (2022) improve on existing results for Vietnamese word segmentation by using an improved LSTM model with a convolutional neural network (CNN) extraction portion.

2.2 Restoration of Capitalization and Punctuation

CRF (Lui and Wang, 2013) and finite state automata (FSA) (Gravano et al., 2009) have been used to restore capitalization and punctuation, but state-of-the-art approaches use neural network models. Che et al. (2016) demonstrate that both deep neural networks (DNNs) and CNNs can outperform CRF-based approaches for restoration of commas, periods, and question marks on a dataset of Ted Talks.

Recent work has demonstrated the advantages of using Transformer architectures for restoration tasks. Nguyen et al. (2019) restore capitalization and punctuation in a single model using a Transformer-enriched sequence-to-sequence LSTM model, while Wang et al. (2018) apply an Encoder-Decoder Transformer architecture with attention to restoration of punctuation only. Yi et al. (2020) and Courtland et al. (2020) achieve state-of-the-art results for punctuation restoration using deep bidirectional Transformer architectures with the pre-trained language models BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019b) respectively. Guerreiro et al. (2021) improve on the work of Yi et al. (2020) by training and evaluating their models at chunk level rather than segment level.

Guerreiro et al. (2021) also successfully train a single model for punctuation restoration in multiple languages using the multilingual language model XLM-RoBERTa (Conneau et al., 2019). However, they observe that better results for English are obtained when using the monolingual RoBERTa. Gupta et al. (2022) use IndicBERT (Kakwani et al., 2020) to carry out punctuation restoration for 11 Indic languages, including Gujarati.

All of the studies cited above adopt purely lexical approaches, but others (Tilk and Alumäe, 2015; Klejch et al., 2017; Yi and Tao, 2019) have incorporated audio features from original ASR inputs.

Guan (2020) proposes an end-to-end ASR system with punctuation restoration included. Zhu et al. (2022) train a BiLSTM model with a hybrid representation to enable restoration of punctuation to unpunctuated texts either with or without accompanying audio.

Gale and Parthasarathy (2017) is a rare example of a character-level approach to punctuation restoration, in which results on-par with state-of-the-art CRF-based models are obtained using various architectures containing LSTM networks. The models in that paper differ from our own in that they restore punctuation only, and each output tag contains only a single feature.

2.3 Restoration of Spaces, Capitalization and Punctuation

The only study of which we are aware that deals with restoration of spaces, punctuation, and capitalization as a single task is Sivakumar et al. (2021). The authors use a single model type— bigram-level GRU—but adopt a pipeline approach where one feature is restored successively in each model component. Interestingly, they find that the best results are obtained by restoring spaces later in the restoration process, after periods and commas but before capitalization.

3 Models

In this section we describe the models used in this study, which encompass a range of both traditional and neural machine learning techniques for restoring spaces, punctuation, and capitalization, or a subset of those features. The model names in **bold** are unique names that we assigned to the models for the purpose of this paper.

NB is a Naive Bayes-based model that uses dynamic programming with memoization to recursively assign probabilities to possible word segmentations based on frequency lists of words learnt during training. Our implementation closely follows the description of the bigram model in Norvig (2009), but we treat the maximum possible word length L and smoothing parameter λ, which are assigned constant values in the original work, as tunable hyperparameters.

Since **NB** assigns probabilities to word segmentations recursively, inference can only be run on string of up to around 100 characters in length due to recursion limits, so longer inputs have to be broken up into shorter chunks. We use the method in Jenks (2018), where the last few words of each chunk is appended to the beginning of the following chunk to prevent words being incorrectly segmented due to being cut off at the end of a chunk. The same method is employed for **BiLSTMCharSpace** and **BiLSTMCharE2E** to allow for control of the model input length without the need for padding tokens.

BiLSTMCharSpace and **BiLSTMCharE2E** are character-level BiLSTM sequence-to-sequence classification models. They were implemented from scratch for this paper and are not based on existing literature. **BiLSTMCharSpace** is for restoration of spaces only, a binary classification task, so binary cross entropy is used for the loss function and sigmoid is used for the activation function. **BiLSTMCharE2E** uses multi-class classification to restore any possible combination of features following each character. The number of possible classes when restoring n features is 2^n, so for our standard set of features for English a maximum of 16 classes are possible. Categorical cross entropy is used for the loss function and softmax is used for the activation function.

Training examples for **BiLSTMCharSpace** and **BiLSTMCharE2E** were generated using a sliding window over each document with a step size of one word (or three characters in the case of the **OshieteQA** dataset). This approach ensures that less frequent words appear multiple times in the training data. The sequence length was set to 200 characters so that at least two or three sentences would be included in each training sample in order to provide sufficient context for restoration of end punctuation.

CRF uses CRF to restore punctuation and capitalization through multi-class token classification. Our implementation is based on Lui and Wang (2013), but where that study assigns only 4 classes to capture presence or absence of any punctuation and/or capitalization, we increase the number of classes to capture presence or absence of a period and/or comma directly after a token and whether a token is lowercase, title cased, or fully capitalized. Also, the previous study uses named entity recognition (NER) results on uncapitalized input text as a model feature, but we skip this step as we found in initial experiments that the low accuracy of NER without capitalization meant that the impact on model performance was minimal.

BERTBiLSTM is a token-level model which

uses a pre-trained Transformer layer, BERT (Devlin et al., 2018), with the final hidden layer attached to a BiLSTM with sigmoid activation. Classes capture combinations of features, for example a period following a token and capitalization of the first letter of the token. Unlike **CRF**, this model does not have a class for full capitalization of a token. It has a maximum sequence length of 256, with a padding token used to fill any remaining slots in each sequence. Padding tokens are masked to avoid the attention layer deriving importance from them. The model is based on the punctuation restoration model in Alam et al. (2020), but expands on that work by restoring capitalization in the same model.

Models for space restoration and models for restoring other features were combined to form pipeline models for restoration of all features. For example, **NB + BERTBiLSTM** refers to the result of inputting the outputs of **NB** into **BERTBiLSTM**. **BiLSTMCharE2E** is the only model that can restore all features in a single inference step.

4 Experiments

4.1 Languages

The languages studied in this paper are English, Japanese, and Gujarati. These languages were selected in order to ascertain the suitability and performance of the novel character-level model **BiLSTMCharE2E** on a diverse range of language types.

English is a spaced language with rich use of capitalization and punctuation. It is also the most widely studied language in the field and its inclusion was essential for comparison with existing work. Japanese is a *scriptio continua* language in which spaces are not inserted between word tokens, and is also in high demand for NLP applications. Gujarati is a low-resource language belonging to the group of Indic languages with special typographical and orthographic characteristics that need to be taken into account when developing character-based models (W3C Group, 2022).

Further, the authoring team contained at least one member who is either native or proficient in each of the languages under study, which made direct qualitative evaluation and analysis of model outputs possible.

4.2 Datasets

In this section we describe the datasets used in this study. The dataset names in **bold** are unique names that we assigned to the datasets for the purpose of this paper.

Our main dataset for English was a corpus of transcripts of TED Talks collected in June 2020 (Gupta, 2020) (hereafter referred to as **TedTalks**). This dataset was chosen because it consists of high-quality professional transcriptions of spoken English, and so closely reflects the intended use case and desired outputs of our models. Ted Talk transcripts have also been used in many prior studies on punctuation and capitalization restoration including Che et al. (2016), Wang et al. (2018), and Courtland et al. (2020). Our aim when preparing this dataset was to establish the feasibility of each of the models under investigation. We therefore carried out thorough data cleaning with the aim of restricting the input vocabulary while retaining as much of the context of the original as possible. For example, we replaced question marks and exclamation marks with periods, "$" signs with the acronym "USD", and "%" signs with the word "percent". We also removed speaker indicators (e.g. "Narrator:") and descriptions of audience activity (e.g. "(Applause)"), which appear frequently in the original corpus. The cleaned dataset consisted of only upper- and lower-case alphabetic characters, digits, spaces, commas, and periods.

The second dataset for English was the **Brown** corpus, which was used for comparison with Sivakumar et al., 2021. Data cleaning was carried out following the procedure described in Section *A.2* of that paper in order to mirror their experiments as closely as possible.

The Japanese and Gujarati datasets were used to ascertain the feasibility of **BiLSTMCharE2E** on languages with larger character vocabularies. Only minimal data cleaning was carried out for these datasets.

For Japanese, we collected our own corpus, **JapaneseQA**, of questions and answers written in 2021 from the popular Q&A site Oshiete! goo (2022) ("教えて！goo"/"Tell me! goo"). The dataset contains questions and answers from a wide range of categories (16 in total), with the most common being "life advice", "education/science/learning" and "health/beauty/fashion". Both full-width and half-width spaces (which do not generally belong in Japanese text) were re-

Dataset	Language	Num. docs	Num. words	Num. chars	Num. unique chars
TedTalks	English	3,997	7,149,001	39,490,824	65
Brown	English	500	1,023,563	5,921,920	65
JapaneseQA	Japanese	42,940	N/A	25,634,557	5,489
GujaratiNews	Gujarati	3,498	906,498	3,814,697	1,470

Table 1: Datasets used in the experiments

moved, as were line breaks after appending a period (“。”) to lines without end punctuation. Exclamation marks (”！”) were replaced with periods. The texts in this dataset are not transcribed from spoken Japanese, but are written in a conversational style that is similar to polite spoken Japanese, and were chosen for this reason. Since they are written by individual site users as opposed to professional writers or transcribers, the punctuation usage is less consistent than for our other datasets.

For Gujarati, we collected our own corpus, **GujaratiNews**, of news articles written in 2021 and 2022 from the news website Gujarat Samachar (2022). We used written data due to the difficulty of obtaining large collections of transcriptions of spoken Gujarati. We collected article headlines and body text from six of the site's category pages. A period and space were appended to lines without end punctuation before removing line breaks.

The numbers in Table 1 describe the size of each dataset before splitting into train and test sets. A randomly selected 20% of the documents in each dataset were saved for testing, with the remaining 80% used for training and validation.

4.3 Evaluation Metrics

A combination of character- and word-level metrics was used to evaluate the results of our experiments. All metrics were calculated by comparing reference documents in the test sets with document-level model outputs (after combining chunks, etc.).

We used character-level metrics—precision, recall, and F-score for each restored feature—as our primary metrics in order to ensure comparability across all experiments. These scores were calculated based on the features possessed by corresponding non-feature characters in the reference and hypothesis outputs. Since our models were not designed to restore multiple instances of the same feature character after a single input character (e.g. `...`), only presence or absence of each feature for each non-feature character was considered. The character-level metrics were highly informative for understanding the performance of models on each restored feature and identifying future areas for improvement. We do not report character-level accuracy because it takes into account true negatives and therefore can be misleadingly high for features that do not occur frequently.

We also provide word error rate (WER) for all experiments in which spaces were among the features restored. Apart from being a standard metric for ASR tasks, this metric has the advantage of summarizing the overall performance in each experiment in a single percentage, and we (like Sivakumar et al., 2021) found that it closely correlated with our subjective evaluation of model outputs based on observation. Reference and hypothesis texts were split at space characters and the Levenshtein distance between them——the sum of substitutions (S), deletions (D), and insertions (I) required to get from the hypothesis to the reference——obtained. Any number of false positives or negatives across the possible features for a single word prompted a single edit (because, for example, `However,` is not the same as either `however,` or `However.,`). WER is calculated as follows:

$$\text{WER } (\%) = \frac{(S + D + I)}{\text{len}_{\text{ref}}} \times 100 \qquad (1)$$

where len_{ref} is the number of words in the reference text.

Lower WERs indicate better model performance, with 0% indicating a perfect match between reference and hypothesis outputs. WER is affected by the relative frequency of restored features in each dataset, so should not be used to compare the results of experiments on different datasets or when restoring different features.

In addition to quantitative metrics, we also found qualitative observation of reconstructed texts to be highly informative, so we include selected extracts from the model outputs to illustrate some of our observations. In all of the sample outputs presented, a dark gray highlight indicates a false positive, while a light gray highlight indicates a false negative. Fea-

Model	Dataset	Units	Batch size	Dropout	Recurrent dropout
BiLSTMCharSpace	**TedTalks**	$\{128, \mathbf{256}\}$	$\{\mathbf{2048}, 4096, 8192\}$	$\{\mathbf{0}, 0.1, 0.2\}$	$\{0, \mathbf{0.1}, 0.2\}$
BiLSTMCharSpace	**Brown**	$\{128, \mathbf{256}\}$	$\{2048, \mathbf{4096}, 8192\}$	$\{0, 0.1, \mathbf{0.2}\}$	$\{\mathbf{0}, 0.1, 0.2\}$
BiLSTMCharE2E	**TedTalks**	$\{128, \mathbf{256}\}$	$\{\mathbf{2048}, 4096, 8192\}$	$\{\mathbf{0}, 0.1, 0.2\}$	$\{\mathbf{0}, 0.1, 0.2\}$
BiLSTMCharE2E	**Brown**	$\{\mathbf{128}, 256\}$	$\{\mathbf{2048}, 4096, 8192\}$	$\{0, 0.1, \mathbf{0.2}\}$	$\{0, \mathbf{0.1}, 0.2\}$
BiLSTMCharE2E	**OshieteQA**	$\{32, 64, \mathbf{128}\}$	$\{\mathbf{128}, 256\}$	$\{\mathbf{0}, 0.1, 0.2\}$	$\{0, 0.1, \mathbf{0.2}\}$
BiLSTMCharE2E	**GujaratiNews**	$\{64, 128, \mathbf{256}\}$	$\{\mathbf{512}, 1024, 2048\}$	$\{\mathbf{0}, 0.1, 0.2\}$	$\{0, \mathbf{0.1}, 0.2\}$

Table 2: Hyperparameters used for **BiLSTMCharSpace** and **BiLSTMCharE2E**

tures without any highlight are true positives—i.e. correctly restored features.

4.4 Hyperparameter Tuning

Tunable hyperparameters for **NB** are L, the maximum possible word length, and λ, the smoothing parameter. Hyperparameters are applied at inference time, so grid search was carried out by running inference on 5% of the test data with each candidate hyperparameter combination and selecting the one with the highest space F-score for the final model. Combinations with $L \in \{14, 16, 18, 20, 22\}$ and $\lambda \in \{6.0, 8.0, 10.0, 12.0, 14.0\}$ were tested. The hyperparameters selected for use in the final models were $L = 16, \lambda = 12.0$ for **TedTalks** and $L = 14, \lambda = 14.0$ for **Brown**.

Tunable hyperparameters for **BiLSTMCharSpace** and **BiLSTMCharE2E** are the number of BiLSTM units, batch size, dropout rate, and recurrent dropout rate. Grid search was carried out for each model/dataset pair by training on 10% of the training data for one epoch with each candidate hyperparameter combination and selecting the one with the highest validation accuracy for use in the final model. Some candidate hyperparameter combinations overloaded GPU memory, which suggests that GPU resources were optimized for successful combinations. Candidate units and/or batch sizes had to be reduced for the Gujarati and Japanese datasets to generate a sufficient number of viable hyperparameter combinations; this is thought to be due to the larger input vocabularies for these models. Candidate hyperparameters for all model/dataset pairs are presented in Table 2, with the hyperparameters used in the final models displayed in **bold**.

CRF has three tunable hyperparameters: C_1, C_2, and the $possible\_transitions$ parameter, which specifies whether to generates transitions between feature pairs that were not present in the training set. We used the values $C_1 = 1.0$, $C_2 = 1\text{e}{-3}$, and $possible\_transitions = True$, following the example notebook by Korobov (2016).

BERTBiLSTM also has three tunable hyperparameters: learning rate, decay and gradient clip. These were set to $5e-6$, 0, and 1 respectively, the same values as used in Alam et al. (2020).

5 Results

All five of our pipeline and end-to-end models were applied to restoration of capitalization, spaces, periods, and commas for the two English-language datasets. The results for **TedTalks** are presented in Table 3. The metrics for pipelines containing **BiLSTMCharSpace** were almost identical to those containing **NB**, with precision, recall, and F-score differing by no more than 0.01 for each feature and WER differing by no more than 0.28%. The results of pipelines containing **BiLSTMCharSpace** are therefore omitted for brevity. The results for the most performant pipeline model and for the end-to-end model for **Brown** are presented in Table 4.

5.1 Space Restoration (English)

Both **NB** and **BiLSTMCharSpace** have F-scores of 0.99 for the space restoration task on the **TedTalks** dataset. The numbers of space restoration errors in the results for each model are very close, but the errors occur in different places. Of the 22 errors in the results for **BiLSTMCharSpace** on the first document in the test set, only 7 were in words where **NB** also made an error. This is consistent with the observation made in Sun (2010), and suggests it may be possible to obtain higher performance by combining the outputs of the two models in some way.

NB assigns probabilities based on whole tokens learnt during training, so errors occur for unlearnt words that can be formed by concatenating two or more learnt words, such as in `un filled` ("unfilled" was not in the training data). **BiLSTMCharSpace** learns at the character level and appears to have implicitly learnt some of the common morphemes in the English language, as it was able to output the correct word in this and similar examples.

NB + CRF
WER: 19.85%

	Precision	Recall	F-score
Spaces (' ')	0.99	0.99	0.99
CAPS	0.74	0.37	0.49
Periods ('.')	0.59	0.34	0.43
Commas (',')	0.46	0.22	0.30
All	**0.95**	**0.86**	**0.90**

NB + BERTBiLSTM
WER: 8.49%

	Precision	Recall	F-score
Spaces (' ')	0.99	0.99	0.99
CAPS	0.88	0.81	0.84
Periods ('.')	0.82	0.82	0.82
Commas (',')	0.77	0.67	0.72
All	**0.96**	**0.95**	**0.95**

BiLSTMCharE2E
WER: 20.48%

	Precision	Recall	F-score
Spaces (' ')	0.98	0.98	0.98
CAPS	0.69	0.62	0.65
Periods ('.')	0.58	0.53	0.56
Commas (',')	0.50	0.38	0.43
All	**0.92**	**0.89**	**0.90**

Table 3: Results from a selection of models on the **TedTalks** dataset.

BiLSTMCharSpace + BERTBiLSTM
WER: 17.27%

	Precision	Recall	F-score
Spaces (' ')	0.97	0.97	0.97
CAPS	0.77	0.73	0.75
Periods ('.')	0.69	0.70	0.69
Commas (',')	0.60	0.40	0.48
All	**0.93**	**0.91**	**0.92**

BiLSTMCharE2E
WER: 24.46%

	Precision	Recall	F-score
Spaces (' ')	0.97	0.97	0.97
CAPS	0.74	0.42	0.54
Periods ('.')	0.55	0.38	0.45
Commas (',')	0.51	0.11	0.17
All	**0.94**	**0.84**	**0.89**

Table 4: Results from the best pipeline model and the end-to-end model on the **Brown** dataset.

Both models tend to commit errors in cases where part of an input string can be segmented into more than one pair of learnt words, such as the input sequence `chargeshe`. **NB** sometimes chooses the wrong segmentation for the context (in this case `charge she`), whereas **BiLSTMCharSpace** sometimes inserts a space in both positions (`charge s he`). The output of **BiLSTMCharSpace** may be easier to post-edit in such cases, as errors would be picked up by a spellchecker.

BiLSTMCharE2E performs slightly worse than **NB** and **BiLSTMCharSpace** in the space restoration task for **TedTalks**, which suggests that a larger number of possible output classes reduces performance on each individual feature. However, the F-score is still very high, at 0.98, which shows that **BiLSTMCharE2E** is able to accurately restore spaces at the same time as other features.

The results of each model for the space restoration task were also very similar for the **Brown** dataset, with pipelines containing **NB** and **BiLSTMCharSpace** both having F-scores of 0.97 for space restoration. However, for the **Brown** dataset, a pipeline containing **BiLSTMCharSpace** had the highest overall F-score, which suggests that **BiLSTMCharSpace** may perform better than **NB** when less training data is available.

5.2 Capitalization and Punctuation Restoration (English)

Pipeline models containing **BERTBiLSTM** outperformed those containing **CRF** on all features. This is thought to be because **BERTBiLSTM** can take into account more contextual information and also benefits from the pre-trained word Transformer layer. The outputs of **NB + BERTBiLSTM** for the **TedTalks** dataset are generally very readable and closely match the punctuation style of the training data. Figure 1 shows a typical extract. The one divergence from the reference document is also a valid punctuation. The outputs of the other models contain more severe errors that result in less readable text.

BiLSTMCharE2E outperformed pipelines containing **CRF** at capitalization and punctuation restoration (likely owing to its ability to take into account a greater amount of contextual information), with a lower overall F-score due solely to poorer performance at space restoration. Combined with its simplicity due to being able to restore all features in a single model, this makes **BiLSTM-**

```
Imagine a brilliant neuroscientist named Mary. Mary lives in a black
 and white room., She only reads black and white books, and her scre
ens only display black and white. But even though she has never seen
```

Figure 1: Output of **NB + BERTBiLSTM** for an extract from the test set for **TedTalks**

WER: N/A

	Precision	**Recall**	**F-score**
Periods (’。’)	0.54	0.53	0.54
Commas (’、’)	0.31	0.31	0.31
Q. marks (’？’)	0.56	0.48	0.52
All	**0.43**	**0.43**	**0.43**

Table 5:
Results from **BiLSTMCharE2E** on **JapaneseQA**

CharE2E a very viable model for the task under consideration. Further, observation of output texts confirms that **BiLSTMCharE2E** is able to correctly restore mid-token punctuation and capitalization. It was the only model able to correctly restore the middle periods in `B.C.` and `D.C.`, and the final-letter capital in `PhD`, to cite a few examples observed in the first 100 test documents.

The gap in performance between the best-performing pipeline model and **BiLSTMCharE2E** was smaller for **Brown** than for **TedTalks**, which suggests that the negative impact on performance from reducing the volume of training data is lower for **BiLSTMCharE2E**.

Based on reading of the paper and consultation with the authors, we believe the "Distance" metric in Sivakumar et al. (2021) to be equivalent to the metric referred to in this paper as WER. Since the lowest Distance reported in that paper at the testing stage is 29.7, we conclude that all of our models (WER 17.27-24.46%) outperform those in that paper. Inspection of the sample outputs presented in that paper compared with those from our models supports this claim.

5.3 Other Languages

Results for **BiLSTMCharE2E** on Japanese and Gujarati datasets demonstrate that this model can handle large input vocabularies, does not depend on extensive data cleaning, and is readily applicable to *scriptio continua* languages.

Results for the **OshieteQA** dataset are presented in Table 5. F-scores for punctuation marks are comparable to those for the same model on the **TedTalks** dataset, and scores of more than 0.5 for both periods and question marks indicate that the model is able to differentiate to a large extent between sentences and questions based on context. The overall F-score is lower because spaces were not among the features considered for Japanese.

Since **JapaneseQA** consists of texts written by non-professional writers, there is a lot of irregular punctuation in the dataset. Inspection of deviations from the reference examples in the model results reveals that in some cases the model actually renders better punctuations than the "gold standard". A similar normalizing effect is noted in Tilk and Alumäe, 2015. In the example in Figure 2, the author has overused commas, and the model left out some of the unnecessary ones. In order to have this verified by expert and impartial judges, 5 native Japanese speakers were given the characters stripped of punctuation and asked to insert commas and periods as they deemed appropriate. The first comma omitted by the model was included by only 3 out of the 5 participants, and the second was not included by any of them. The irregularity of the dataset may have led to the quantitative metrics underestimating the model’s performance.

Initial results for Gujarati revealed a flaw in the implementation of **BiLSTMCharE2E**. Our original implementation split input strings at the byte level for training and inference, but Gujarati differs from English and Japanese in that graphemes are not equivalent to bytes because combinations of vowel and consonant characters can form a single grapheme. While the model was able to learn to some extent from byte-level information, space restoration performance was surprisingly low (F-score 0.69), and incorrectly restored spaces sometimes caused characters intended to be displayed together with another character as a single grapheme to be displayed separately, leading to ugly and unreadable outputs. This issue was resolved by splitting strings into grapheme clusters rather than bytes. Grapheme clusters are equivalent to bytes for most languages, so the improved model implementation still works as intended for those languages.

Results for **GujaratiNews** when **BiLSTMCharE2E** is trained on grapheme clusters are presented in Table 6. The F-score for spaces is close to those for the English datasets, and F-scores for

ＢＤについては、[3]リージョンＡで、[3]北米も同じだから、[2]再生も可ってことになる。
ＤＶＤビデオの場合だと、[5]プレ イヤーによっては、[0]リージョンフリーのものもある。

For BD it's region A, the same as North America, so playback is possible.
In the case of DVD video, some players are region-free.

Figure 2: An extract from the results of **BiLSTMCharE2E** for the first document in the test set for **JapaneseQA**. The numbers in superscript indicate how many out of a sample of 5 native Japanese speakers included each comma.

WER: 13.85%

	Precision	**Recall**	**F-score**
Spaces (' ')	0.97	0.97	0.97
Periods ('.')	0.86	0.81	0.83
Commas (',')	0.57	0.45	0.50
All	**0.96**	**0.94**	**0.95**

Table 6:
Results from **BiLSTMCharE2E** on **GujaratiNews**

periods and commas are the highest of all of the datasets. This may be due partly to greater regularity of the punctuation usage in the dataset due to its genre, and regularity in the characters used to end clauses and sentences in Gujarati. The scores are nonetheless impressive given the relatively small size of the dataset.

6 Conclusion and Future Work

We assessed a variety of pipeline and end-to-end models for restoration of spaces, punctuation and capitalization on unformatted English, Japanese, and Gujarati texts. Of all of the models considered in this study, the pipeline model **NB + BERTBiLSTM** outperforms the others for restoration of all features under consideration. Both components of this pipeline model take into account token-level information, which suggests that there are advantages to using this information over purely character-based approaches. In particular, a neural network model with a pre-trained Transformer layer was highly performant in restoration of capitalization and punctuation, allowing the pipeline model to render readable outputs that could be used in the real world with very minimal post-editing. **BERTBiLSTM** could be improved further by including a class to restore full capitalization to a token to improve scores for capitalization restoration.

The end-to-end character-based model **BiLSTMCharE2E** outperformed pipeline models containing a CRF-based component for punctuation and capitalization restoration, and only slightly underperformed those models for space restoration. Furthermore, **BiLSTMCharE2E** was shown to be easily applicable to different languages and sets of features, and to be able to restore features to individual characters inside tokens.

Due to the observed advantages of using pre-trained word embeddings for token-level models, we are very interested in investigating the effect of using pre-trained byte level language models such as ByT5 (Xue et al., 2021) on both character- and token-level models. This is left for future work.

References

Samia Abd-hood and Nazlia Omar. 2021. Hashtag segmentation: A comparative study involving the Viterbi, triangular matrix and word breaker algorithms. *Journal of Advances in Information Technology*, 12:311–318.

Tanvirul Alam, Akib Khan, and Firoj Alam. 2020. Punctuation restoration using transformer models for high- and low-resource languages. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 132–142, Online. Association for Computational Linguistics.

Xiaoyin Che, Cheng Wang, Haojin Yang, and Christoph Meinel. 2016. Punctuation prediction for unsegmented transcript based on word vector. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 654–658, Portorož, Slovenia. European Language Resources Association (ELRA).

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *Computing Research Repository*, arXiv:911.02116. Version 2.

Maury Courtland, Adam Faulkner, and Gayle McElvain. 2020. Efficient automatic punctuation restoration using bidirectional transformers with robust inference. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 272–279, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language

understanding. *Computing Research Repository*, arXiv:1810.04805. Version 2.

William Gale and Sarangarajan Parthasarathy. 2017. Experiments in character-level neural network models for punctuation. In *Interspeech 2017*, pages 2794–2798.

Agustin Gravano, Martin Jansche, and Michiel Bacchiani. 2009. Restoring punctuation and capitalization in transcribed speech. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4741–4744, Taipei, Taiwan. IEEE.

Yushi Guan. 2020. End to end ASR system with automatic punctuation insertion. *Computing Research Repository*, arXiv:2012.02012.

Nuno Miguel Guerreiro, Ricardo Rei, and Fernando Batista. 2021. Towards better subtitles: A multilingual approach for punctuation restoration of speech transcripts. *Expert Systems with Applications*, 186:115740.

Gujarat Samachar. Gujarat Samachar [online]. 2022. [Accessed 31 July 2022].

Anirudh Gupta, Neeraj Chhimwal, Ankur Dhuriya, Rishabh Gaur, Priyanshi Shah, Harveen Singh Chadha, and Vivek Raghavan. 2022. indic-punct: An automatic punctuation restoration and inverse text normalization framework for Indic languages. arXiv:2203.16825. [Submitted to InterSpeech 2022].

Vishal Gupta. TED Talk [online]. 2020. [Accessed 31 July 2022].

Choochart Haruechaiyasak, Sarawoot Kongyoung, and Matthew Dailey. 2008. A comparative study on Thai word segmentation approaches. In *5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, volume 1, pages 125–128, Krabi, Thailand. IEEE.

Shohei Higashiyama, Masao Utiyama, Eiichiro Sumita, Masao Ideuchi, Yoshiaki Oida, Yohei Sakamoto, and Isaac Okada. 2019. Incorporating word attention into character-based word segmentation. 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 2699–2709, Minneapolis, Minnesota. Association for Computational Linguistics.

Grant Jenks. Python word segmentation [online]. 2018. [Accessed 31 July 2022].

Douglas Jones, Florian Wolf, Edward Gibson, Elliott Williams, Evelina Fedorenko, Douglas Reynolds, and Marc Zissman. 2003. Measuring the readability of automatic speech-to-text transcripts. In *8th European Conference on Speech Communication and Technology (Eurospeech 2003)*, pages 1585–1588.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Ondřej Klejch, Peter Bell, and Steve Renals. 2017. Sequence-to-sequence models for punctuated transcription combining lexical and acoustic features. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5700–5704.

Mikhail Korobov. python-crfsuite [online]. 2016. [Accessed 20 August 2022].

Junxin Liu, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2019a. Neural Chinese word segmentation with dictionary. *Neurocomputing*, 338:46–54.

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

Marco Lui and Li Wang. 2013. Recovering casing and punctuation using conditional random fields. In *Proceedings of the Australasian Language Technology Association Workshop 2013 (ALTA 2013)*, pages 137–141, Brisbane, Australia.

Stephen Mayhew, Tatiana Tsygankova, and Dan Roth. 2019. ner and pos when nothing is capitalized. *Computing Research Repository*, arXiv:2012.02012. Version 2.

Binh Nguyen, Vu Bao Hung Nguyen, Hien Nguyen, Pham Ngoc Phuong, The-Loc Nguyen, Quoc Truong Do, and Luong Chi Mai. 2019. Fast and accurate capitalization and punctuation for automatic speech recognition using transformer and chunk merging. In *2019 22nd Conference of the Oriental COCOSDA International Committee for the Co-ordination and Standardisation of Speech Databases and Assessment Techniques (O-COCOSDA)*, pages 1–5, Cebu, Philippines. IEEE.

Thai-Son Nguyen, Sebastian Stüker, and Alex Waibel. 2020. Super-human performance in online low-latency recognition of conversational speech. *Computing Research Repository*, arXiv:2010.03449. Version 5.

Peter Norvig. 2009. Natural language corpus data. In Toby Seagaran and Toby Hammerbacher, editors, *Beautiful Data*, pages 219–242. O'Reilly, Sebastopol, Canada.

Oshiete! goo. Oshiete! goo [online]. 2022. [Accessed 31 July 2022].

Stephan Peitz, Markus Freitag, Arne Mauser, and Hermann Ney. 2011. Modeling punctuation prediction as machine translation. In *International Workshop on Spoken Language Translation*, pages 238–245, San Francisco, California.

Yan Shao, Christian Hardmeier, and Joakim Nivre. 2018. Universal word segmentation: Implementation and interpretation. *Transactions of the Association for Computational Linguistics*, 6:421–435.

Jasivan Sivakumar, Jake Muga, Flavio Spadavecchia, Daniel White, and Burcu Can. 2021. A GRU-based pipeline approach for word-sentence segmentation and punctuation restoration in English. In *2021 International Conference on Asian Language Processing (IALP)*, pages 268–273.

Savina van der Straten. Voice tech landscape: 150+ infrastructure, horizontal and vertical startups mapped and analysed [online]. 2017. [Accessed 31 July 2022].

Weiwei Sun. 2010. Word-based and character-based word segmentation models: Comparison and combination. In *Coling 2010: Posters*, pages 1211–1219, Beijing, China. Coling 2010 Organizing Committee.

Gabriel Synnaeve. wer_are_we [online]. 2022. [Accessed 31 July 2022].

Ottokar Tilk and Tanel Alumäe. 2015. LSTM for punctuation restoration in speech transcripts. In *Interspeech 2015*, pages 683–687.

Lynne Truss. 2004. *Eats, Shoots & Leaves: the Zero Tolerance Approach to Punctuation*. Gotham Books, New York, New York.

W3C Group. Gujarati Gap Analysis [online]. 2022. Accessed 3 Nov 2022].

Feng Wang, Wei Chen, Zhen Yang, and Bo Xu. 2018. Self-attention based network for punctuation restoration. In *24th International Conference on Pattern Recognition (ICPR)*, pages 2803–2808, Beijing, China. IEEE.

Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. 2016. Achieving human parity in conversational speech recognition. *Computing Research Repository*, arXiv:1610.05256. Version 2.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Computing Research Repository*, arXiv:2105.13626. Version 3.

Jiangyan Yi and Jianhua Tao. 2019. Self-attention based model for punctuation prediction using word and speech embeddings. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7270–7274.

Jiangyan Yi, Jianhua Tao, Ye Bai, Zhengkun Tian, and Cunhang Fan. 2020. Adversarial transfer learning for punctuation restoration. *Computing Research Repository*, arXiv:2004.00248.

Qi Zhang, Xiaoyu Liu, and Jinlan Fu. 2018. Neural networks incorporating dictionaries for Chinese word segmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Kexiao Zheng and Wenkui Zheng. 2022. Deep neural networks algorithm for Vietnamese word segmentation. *Scientific Programming*, 2022.

Yaoming Zhu, Liwei Wu, Shanbo Cheng, and Mingxuan Wang. 2022. Unified multimodal punctuation restoration framework for mixed-modality corpus. *Computing Research Repository*, arXiv:2202.00468.

New Features for Discriminative Keyword Spotting

Punnoose A K
Flare Speech Systems
Bangalore, India

Abstract

This paper presents new feature functions and an efficient training approach to discriminative keyword spotting. A 5-dimensional feature function is derived from a frame-based phoneme classifier and a pitch detector. The mechanism by which each feature function finds a correspondence between the keyword-specific variables and the speech segment is discussed. Multiple aspects of the keyword, that the feature functions capture, are explored. The keyword-scoring function along with the positive and negative data associated with a keyword is defined. A computationally intensive operation in keyword training is identified and an approach is developed to reduce the training more tractable. The proposed approach is implemented with a set of 10 keywords and benchmarked against a traditional lattice-based keyword search on a real-world dataset, and the results are discussed.

1 Introduction

Keyword spotting refers to the detection of specific words in a speech stream. This is different from automatic speech recognition(ASR) in the sense that models are built exclusively for the keywords. Keyword spotting is used in large-scale audio indexing and retrieval, wake-up word for devices, etc. Traditionally hidden Markov model(HMM) based approaches have been used in keyword spotting (Bahl et al., 1986; Rohlicek et al., 1989; Rose and Paul, 1990; Szöke et al., 2005). Keywords are modelled using keyword-specific HMMs and the rest of the speech is modelled using background models (Wilpon et al., 1990). The likelihood ratio of a speech segment running through a keyword HMM to the background speech HMM is used to detect the keyword. Phone lattice keyword search (Young et al., 1997) searches for the sequence of phonemes of the keyword in the phone lattice generated during N-best Viterbi decoding. The lattice search can be improved by dynamic programming and minimum edit distance (James and Young, 1994; Thambiratnam and Sridharan, 2005). Optimizations like unique arc per-second pruning and posting-list merging (Gales et al., 2017) are employed to prune the lattice for faster keyword search, especially in low-resource keyword spotting. In a low-resource environment, the lattice search has been further optimized using web data for language model training (Mendels et al., 2015) and stimulated training (Ragni et al., 2017).

A phone lattice can be rescored using different approaches to make the path scores more relevant. Acoustic word embeddings (Piunova et al., 2019), lattice context information (Chen and Wu, 2017), word-burst (Ma et al., 2014) have been used to rescore the lattices. Future word contexts have been exploited using recurrent neural language models to rescore the lattices (Chen et al., 2019). Rather than using a single feature, multiple features like hierarchical bottleneck features (Riedhammer et al., 2013), smoothened posteriors (Chen et al., 2014), and multilingual bottleneck features (Menon et al., 2018b) are also used for keyword spotting. To increase robustness, especially in noisy and channel degraded environments, feature fusion (Mitra et al., 2014) is used. Generally, the keyword spotter works on top of ASR. Recently, ASR-free approaches (Menon et al., 2018b; Audhkhasi et al., 2017; Menon et al., 2018a) are also proposed for keyword spotting. This is especially useful in low-resource settings.

Neural networks are often used in conjunction with HMMs for keyword spotting (Rath et al., 2014; Chen and Lee, 2013). Different neural network architectures like deep neural networks (Chen et al., 2014), convolutional neural networks (Sainath and Parada, 2015), compressed time delay neural networks (Sun et al., 2017), recurrent networks (Li et al., 1992; Fernández et al., 2007), long short-term memory(lstm) (Wollmer et al., 2009) have

been employed for keyword detection in speech. Recently transformers have been used for keyword spotting in speech (Berg et al., 2021). In this paper, we follow the framework defined in (Keshet et al., 2009), where a set of feature functions are defined and a weight vector, corresponding to each keyword, is trained in an online discriminative manner. While training, a minimum margin is specified between the keywords and all the other words. In the original setting, the authors derive most of the feature functions directly from the spectral level features, while we use an intermediate layer of a frame classifier and a pitch detector, sitting atop the spectral layer, and deriving feature functions out of it.

In section 2, we review the discriminative keyword spotting framework. The training algorithm, feature functions, and positive and negative vectors are defined. A computationally efficient approach to reduce the individual keyword training time is discussed in detail. In section 3, the experimental details of benchmarking the proposed approach against a traditional lattice-based keyword search are discussed. Section 4 concludes the paper.

2 Problem Setting

First, we define an acoustic tuple (X, P, F) to be a set of vectors derived from passing a speech segment through a frame classifier and a pitch detector. X is the decoded phoneme string sequence with softmax probability sequence P for an utterance of length T frames. F is the sequence of pitch values. A keyword k is associated with a set of positive acoustic tuples $(X_{\mathrm{k}}^{+}, P_{\mathrm{k}}^{+}, F_{\mathrm{k}}^{+})$ and negative acoustic tuples $(X_{\mathrm{k}}^{-}, P_{\mathrm{k}}^{-}, F_{\mathrm{k}}^{-})$. The superscript $+$ implies the presence of the keyword in the speech segment and $-$ indicates its absence. Next, we define a feature function ϕ that takes the form

$$\phi : (V_{\mathrm{k}}, X, P, F) \rightarrow R^5 \tag{1}$$

A feature function broadly takes 2 classes of arguments and maps them into a feature space. V_{k} is associated with the orthographic representation of the keyword k and (X, P, F) is associated with a given speech segment. V_{k} involves the variables like the distribution of the ideal duration of the keyword, dictionary entries of the keyword, number of voiced segments in the keyword, distribution of the ideal duration of the voiced segments in the keyword, etc. The feature function must output similar vector values in some distance sense for the same keyword with similar speech segments. The scoring function for keyword k takes the form

$$f = w_{\mathrm{k}}.\phi(V_{\mathrm{k}}, X, P, F) \tag{2}$$

where w_{k} is the keyword-specific weight vector.

2.1 Training

Given a set of positive and negative acoustic tuples, we wish to learn the weight vector w_{k} for the keyword k. The algorithm operates in an online manner. Let $w_{\mathrm{k}_{i-1}}$ be the weight at the $i-1$th iteration. To find w_{k_i}, we first calculate $(X_{\mathrm{k}}^{*}, P_{\mathrm{k}}^{*}, F_{\mathrm{k}}^{*})$ as,

$$(X_{\mathrm{k}}^{*}, P_{\mathrm{k}}^{*}, F_{\mathrm{k}}^{*}) = \max_{(X_{\mathrm{k}}, P_{\mathrm{k}}, F_{\mathrm{k}})} w_{\mathrm{k}_{i-1}}.\phi(V_{\mathrm{k}}, X_{\mathrm{k}}^{-}, P_{\mathrm{k}}^{-}, F_{\mathrm{k}}^{-}) \tag{3}$$

$(X_{\mathrm{k}}^{*}, P_{\mathrm{k}}^{*}, F_{\mathrm{k}}^{*})$ is the worst possible negative acoustic tuple for w_{k} at the $i-1$th stage. In the i-th step, this has to be penalized. Define Δ_i as,

$$\Delta_i = \frac{1}{|X^{+}||X^{-}|}\left[\phi(V_{\mathrm{k}}, X_{\mathrm{k}}^{+}, P_{\mathrm{k}}^{+}, F_{\mathrm{k}}^{+}) - \phi(V_{\mathrm{k}}, X_{\mathrm{k}}^{*}, P_{\mathrm{k}}^{*}, F_{\mathrm{k}}^{*})\right] \tag{4}$$

To find w_{k_i} from $w_{\mathrm{k}_{i-1}}$, the following optimization problem has to be solved.

$$\begin{aligned} w_{\mathrm{k}_i} = \min_{w} & ||w - w_{\mathrm{k}_{i-1}}|| \\ \text{s.t.} \;\; & w.\Delta_i \geq 1 \end{aligned} \tag{5}$$

The solution to this optimization problem is

$$w_{\mathrm{k}_i} = w_{\mathrm{k}_{i-1}} + \delta_i \Delta_i \tag{6}$$

$$\text{where } \delta_i = min\left\{A, \frac{1 - w_{\mathrm{k}_{i-1}}.\Delta_i}{||\Delta_i||^2}\right\} \tag{7}$$

where A is a complexity-accuracy tradeoff parameter. The optimization in equation (5) ensures that the new weight vector $w_{\mathrm{k}_{i-1}}$ is close to the current weight vector w_{k_i}, yet obeying the margin requirement. The correctness of this online update is proved in (Crammer et al., 2006).

In the original setting (Keshet et al., 2009), the authors use feature functions that map a tuple of an acoustic vector, a phoneme sequence, and an alignment sequence to a real vector. For a keyword, each training unit consists of the phoneme sequence corresponding to the keyword along with a positive and negative acoustic vector. At any stage in training, the worst possible acoustic negative tuple is

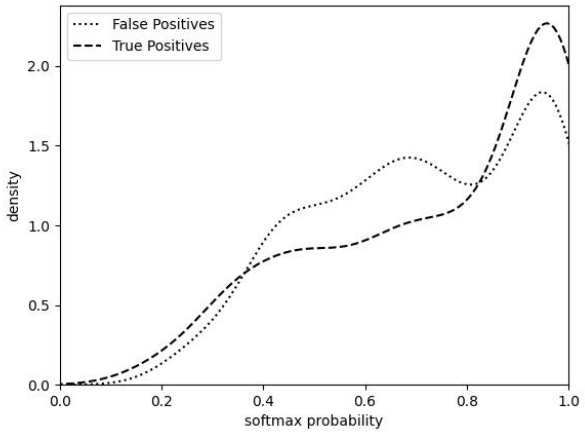


Figure 1: Density of softmax probability of /f/

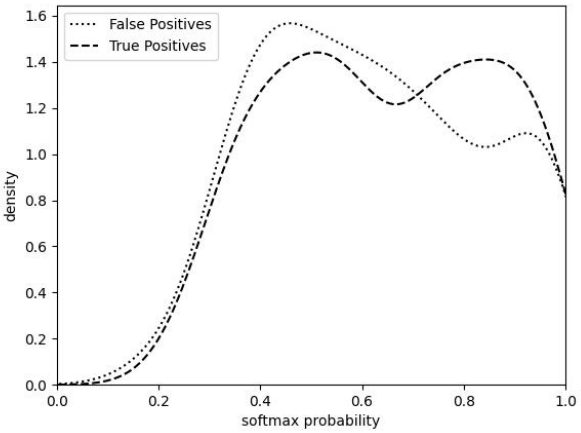


Figure 2: Density of softmax probability of /aa/

the tuple that consists of the keyword phoneme sequence, negative acoustic vector, and the worst possible alignment sequence. In our framework, the acoustic tuples and the feature functions inherently capture the alignment sequence, so that we can get away without having an explicit alignment sequence variable.

2.2 Feature Functions

We define foreign frames of a keyword as the frames that belong to the phonemes which do not fall in any dictionary entry of the keyword. ϕ_1 is the percentage of relevant phonemes detected in sequence in X, in the total number of phonemes in the dictionary entry of the keyword, expressed in decimal. ϕ_1 captures the relevance of phonemes in the speech segment to the keyword. ϕ_2 is the percentage of the relevant frames detected in sequence, in the total number of frames in X, expressed in decimal.

ϕ_3 is based on the observation that, for a frame based phoneme classifier with a softmax output, the density of softmax probability for true positive and false positive phoneme detections are different for different phonemes. ϕ_3 is the average ratio of the density of frame softmax probability of true positives to the false positives of the frames detected correctly in sequence. Fig.1 and Fig.2 plots the density of the softmax probability of true positive and false positive detections of the phonemes /f/ and /aa/. The plot is generated from the frame labelled data d_{train2}, which is explained in section 3. At the peak, the density of true positives is greater than that of false positives. If a set of consecutive frames are detected as /f/, and if all the frames have a softmax probability greater than 0.95, it is more probable that all the frames are true positives. ϕ_3 is calculated as follows.

$$\phi_3(V_{\mathrm{k}}, X, P, F) = \begin{cases} \frac{1}{N}\sum_i \frac{f_{\mathrm{tp}}(p_i;x_i)}{f_{\mathrm{fp}}(p_i;x_i)} & 50\% \text{ rec.} \\ 0 & \text{else} \end{cases} \tag{8}$$

where $f_{\mathrm{tp}}(x_i; p_i)$ and $f_{\mathrm{fp}}(x_i; p_i)$ are the densities of the softmax probability p_i evaluated on the true positive and the false positive softmax probability density curve of the phoneme x_i. N is the number of relevant frames detected in sequence. ϕ_3 returns 0 if less than 50% phonemes in the best possible dictionary entry of the keyword are detected in sequence.

ϕ_4 captures how well the total duration of the detected voiced regions in the speech segment matches the ideal total voiced duration of the keyword. The ideal total voiced duration of a keyword is the sum of all the voiced durations in the keyword and is represented by a normal density, $\mathcal{N}(\mu_{\text{voiced}}, \sigma^2_{\text{voiced}})$. Likewise, ϕ_5 captures how well the duration between the voiced regions at the starting and the ending of the speech segment matches the ideal duration between the voiced regions at the boundaries. ϕ_5 is also represented by a normal density $\mathcal{N}(\mu_{\text{between voiced}}, \sigma^2_{\text{between voiced}})$. ϕ_5 in a sense captures the length of the keyword. ϕ_5 mandates the keywords to have atleast 2 voiced segments. If there are no 2 voiced regions detected in a speech segment, ϕ_5 outputs 0.

2.3 Negative Data

Negative data corresponds to the speech segments where a given keyword is absent. For a big dataset,

the number of speech segments that can be selected where a particular keyword is absent is enormous. We employ a simple heuristic to cut short the number of such speech segments. We compute the average duration d of a keyword from the positive instances of that keyword. From an arbitrary starting frame t, all the speech segments from $(t, t + d/2)$ to $(t, t + 3d/2)$ are treated as separate negative instances of the keyword k, and the corresponding negative acoustic tuples are computed. This makes it extremely skewed to negative instances compared to the positive instances of a keyword.

For a keyword, the calculation of the worst possible acoustic tuple involves the dot product computation of all the negative instances with the present weight vector w as expressed in equation (3). This is computationally expensive as the number of negative instances is often much more than the number of positive training instances of a keyword. To solve this problem, we first compute the convex hull of all the negative vectors of a keyword. Then, take the dot product of the current weight vector with all the extremal points in the convex hull and assign the extremal point with the highest dot product as the worst possible acoustic tuple for the current weight vector.

2.4 Analysis

Consider a finite set of points P in R^d. Let C be the convex hull constructed on the set P. Let E be the set of extremal points of the convex hull. $E \subset P$. Let x be a point outside C. Let $\cdot$ denote the dot product.

Proposition 1 *There exists atleast one point* $e \in E$ *with the condition,* $x \cdot e \geq x \cdot i \quad \forall\, i \in P - E$.

Proof Let us consider R^2. Let b be a point in C such that $x \cdot b \geq x \cdot p$. Further, there are 2 cases.

1. $b \in E$. In this case, Proposition 1 is proved.

2. $b \notin E$. In this case, b is a point in a line segment $\overline{b_1 b_2}$ with endpoints in E. Assume a hyperplane, $H = \{z | x \cdot z = x \cdot p\}$. In R^2, H is a line. We loosely label a point c inside H, if $x \cdot c < x \cdot p$. Now there are 4 possible cases.

   (a) The line segment $\overline{b_1 b_2}$ is such that $x \cdot b_1 > x \cdot p$ and $x \cdot b_2 > x \cdot p$. Both b_1 and b_2 are outside points. Proposition 1 stands true.

   (b) The line segment $\overline{b_1 b_2}$ intersects H. For b_1,

   $$\begin{aligned} & x \cdot b_1 = x \cdot p + x \cdot (b_1 - p) \\ & x \cdot (b_1 - p) < 0 \qquad b_1 \text{ is inside } H \\ & x \cdot b_1 < x \cdot p \end{aligned} \tag{9}$$

   Similarly for b_2,

   $$\begin{aligned} & x \cdot b_2 = x \cdot p + x \cdot (b_2 - p) \\ & x \cdot (b_2 - p) > 0 \qquad b_2 \text{ is outside } H \\ & x \cdot b_2 > x \cdot p \end{aligned} \tag{10}$$

   Proposition 1 stands true.

   (c) Same case as above, where b_2 is inside H and b_1 outside H. Proposition 1 stands true.

   (d) Both $x \cdot b_2 < x \cdot p$ and $x \cdot b_1 < x \cdot p$. i.e, both b_1 and b_2 are inside H. If this is the case,

   $$x \cdot b = x \cdot [\lambda b_1 + (1 - \lambda) b_2] \tag{11}$$
   $$= \lambda x \cdot b_1 + (1 - \lambda) x \cdot b_2 \tag{12}$$
   $$< x \cdot p \tag{13}$$

   which is against the initial assumption $x \cdot b \geq x \cdot p$. Hence this case is infeasible.

Note that the convex hull computation is a one-time operation for a keyword, and is computed in the feature space. Once the convex hull of the negative points of a keyword, i.e, E is computed, the equation (3) becomes

$$\begin{aligned} & (X_{\mathrm{k}}^*, P_{\mathrm{k}}^*, F_{\mathrm{k}}^*) = \\ & \max_{(X_{\mathrm{k}}, P_{\mathrm{k}}, F_{\mathrm{k}})} w_{\mathrm{k}_{i-1}} . \phi(V_{\mathrm{k}}, X_{\mathrm{k}}^-, P_{\mathrm{k}}^-, F_{\mathrm{k}}^-) \\ & \text{where } \phi(V_{\mathrm{k}}, X_{\mathrm{k}}^-, P_{\mathrm{k}}^-, F_{\mathrm{k}}^-) \in E \end{aligned} \tag{14}$$

3 Experimental Details & Results

Voxforge dataset (Voxforge.org) is used in the experiments. The Voxforge dataset is a real-world non-curated dataset that perfectly captures the real-world noise and the acoustic characteristics of different recording equipment. Approximately 40 hours of data(d_{train1}) is forced aligned using Kaldi (Povey et al., 2011) and a baseline multilayer perceptron(MLP) frame classifier, with the architecture 351x1000x1000x1000x41, is built using ICSI Quicknet (Johnson) with perceptual linear prediction(plp) coefficients as the feature input. Standard English phonemes are used as the target. Given a

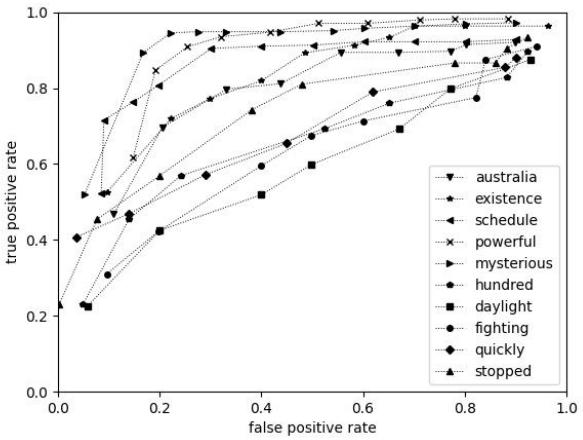


Figure 3: ROC of discriminative keyword spotting

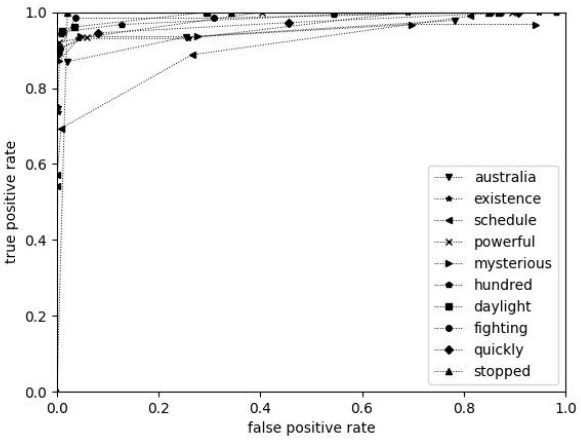


Figure 4: ROC of lattice keyword spotting

9 frame plp input, with each frame corresponding to 25ms with 15ms overlap, the classifier outputs a probability vector of size 41, each component corresponding to a phoneme, and the phoneme corresponding to the highest probability is treated as the recognized phoneme. For identifying the voiced regions, an autocorrelation-based pitch detector (Huckvale) is used. A pitch range of 40-600Hz is used and any value outside is discarded. Pitch values are computed on a 50ms time window and boundary adjusted with the output of the frame classifier. A pitch segment with a pitch difference between the adjacent frames within a 20Hz threshold is treated as a voiced region.

A separate 20 hours of data d_{train2} is used to learn the parameters in ϕ_3, ϕ_4 and ϕ_5. This goes into the keyword-specific variable V_k. Another 20 hours of speech data is run through the baseline classifier and the pitch detector to get the d_{meta}. Keyword present areas in d_{meta} are manually labelled at the word level, and the positive and negative training segments (X, P, F) of each keyword are generated. The positive and negative segments of each keyword are converted to points in the feature space. The convex hull is computed for all the negative data points and the weight vector is trained as described in subsection 2.1 for all the keywords. The value of the complexity-accuracy tradeoff parameter A is set to 1. Ten keywords are chosen, depending on the frequency, duration, and presence of voiced regions in the boundaries. Table 1 shows the number of extremal points on the convex hull computed from a random sample of 20000 points of negative instances of each keyword. Although the number of extreme points of the convex hull of a set is dependent on how the points are distributed, from Table 1, it is clear that atleast an order of magnitude reduction in the computation of dot product is attained. The convex hull is computed as explained in (Barber et al., 1996).

Table 1: Number of extremal points on the convex hull

Keyword	# Extremal Points
australia	126
existence	133
schedule	178
powerful	124
mysterious	204
hundred	172
daylight	264
fighting	151
quickly	181
stopped	152

20 hours of data d_{test} is used for testing. The occurrence of all the keywords is manually time labelled and positive and negative points are generated for all keywords in the feature space. Speech segments, for creating the negative points in the feature space, are generated in the manner specified in subsection 2.4. The scoring function for a speech instance against a keyword takes the form as expressed in equation (2).

The discriminative keyword spotter is benchmarked against a lattice-based keyword search using Kaldi. Kaldi is trained using the same d_{train} and is decoded into a lattice using d_{test}. Multiple phoneme paths through the lattice are searched for keywords. While searching through the lat-

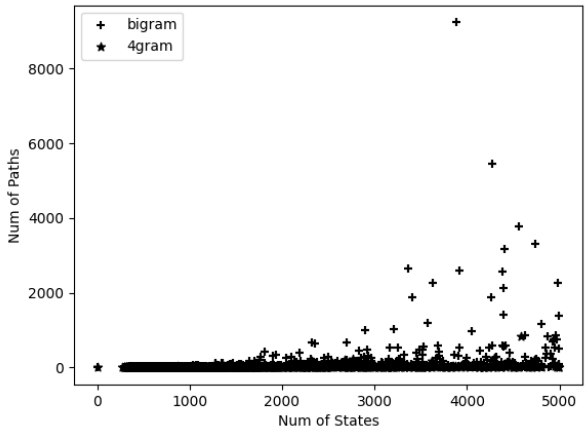


Figure 5: Number of states vs number of all possible paths in the lattice

tice, constraints like reducing the number of cycles and pruning the maximum length of paths, are employed to reduce the search space. We discard the lattice with states more than 500000. The acoustic score in the lattice is ignored. A bigram language model is used for decoding. The phoneme path through the lattice is divided into chunks of the size of the dictionary entry, and the length of the longest subsequence with the dictionary entry of the keyword is found. We consider a keyword to be detected if the longest subsequence length between a phoneme chunk in the lattice path and the dictionary entry is above a certain number of phonemes.

Fig.3 and Fig.4 plots the ROC curve of the discriminative keyword spotter and the lattice keyword spotter for various keywords. The number of phonemes detected in sequence is used as the threshold in Fig.4. It is clear that lattice keyword spotting is superior compared to discriminative keyword spotting. Lattice keyword spotting operates with a language model, while the discriminative spotter is purely acoustic in nature. Moreover, the number of possible paths through a lattice can be large, whereas the discriminative keyword spotter selects speech segments linearly in a recording for locating the keywords. Fig.5 plots the number of all possible paths against the number of states in the lattice, for a small subset of testing data. The maximum number of possible states is limited to 5000. As the number of states in the lattice increases, the number of possible paths also increases. A lattice decoded with a 4-gram language model constrains the number of possible paths compared to that with a bigram.

4 Conclusion

A set of 5 robust feature functions derived from a frame classifier and a pitch detector, for the discriminative approach to the keyword spotting problem, is presented. The feature functions along with the positive and negative training instances of the keyword are defined. The mechanism by which each feature function finds a correspondence between the keyword-specific variables and the speech segment is discussed in detail. Multiple aspects of the keyword, that the feature functions capture, are explored. A computationally intensive operation in keyword training is identified. An approach that makes keyword training more efficient, is presented, proved, and discussed in detail. The proposed approach is implemented with a set of 10 keywords with the Voxforge dataset. To benchmark our approach, a lattice-based keyword search is implemented in Kaldi with the same dataset and the results are compared. ROC curves are plotted for each keyword separately.

The approach shows how feature functions derived from multiple aspects of speech, can be combined to predict the keyword. In the future, the same framework can be used to incorporate more features like formants, acoustic-phonetic characteristics, phoneme-specific spectrogram features, etc for better keyword recognition.

References

Kartik Audhkhasi, Andrew Rosenberg, Abhinav Sethy, Bhuvana Ramabhadran, and Brian Kingsbury. 2017. End-to-end asr-free keyword search from speech. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1351–1359.

Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. 1986. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Proc. ICASSP*, volume 11, pages 49–52.

C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483.

Axel Berg, Mark O'Connor, and Miguel Tairum Cruz. 2021. Keyword transformer: A self-attention model for keyword spotting. In *Interspeech 2021*. ISCA.

Guoguo Chen, Carolina Parada, and Georg Heigold. 2014. Small-footprint keyword spotting using deep neural networks. In *Proc. ICASSP*, pages 4087–4091.

I.-F Chen and Chin-Hui Lee. 2013. A hybrid hmm/dnn approach to keyword spotting of short words. In *Proc. INTERSPEECH*, pages 1574–1578.

Xie Chen, Xunying Liu, Yu Wang, Anton Ragni, Jeremy H. M. Wong, and Mark J. F. Gales. 2019. Exploiting future word contexts in neural network language models for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(9):1444–1454.

Zhipeng Chen and Ji Wu. 2017. A rescoring approach for keyword search using lattice context information. In *Proc. INTERSPEECH*, pages 3592–3596.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. 2007. An application of recurrent neural networks to discriminative keyword spotting. In *Proc. ICANN*.

M.J.F. Gales, Kate Knill, and Anton Ragni. 2017. Low-resource speech recognition and keyword-spotting. pages 3–19.

Mark Huckvale. Ampitch. `https://www.speechandhearing.net/laboratory/ampitch/`.

D.A. James and S.J. Young. 1994. A fast lattice-based approach to vocabulary independent wordspotting. In *Proc. ICASSP*, volume i, pages I/377–I/380.

D Johnson. Icsi quicknet software package. `http://www.icsi.berkeley.edu/Speech/qn.html`.

Joseph Keshet, David Grangier, and Samy Bengio. 2009. Discriminative keyword spotting. *Speech Communication*, 51(4):317–329.

K.P. Li, J.A. Naylor, and M.L. Rossen. 1992. A whole word recurrent neural network for keyword spotting. In *Proc. ICASSP*, volume 2, pages 81–84.

Min Ma, Justin Richards, Victor Soto, Julia Hirschberg, and Andrew Rosenberg. 2014. Strategies for rescoring keyword search results using word-burst and acoustic features. In *Proc. INTERSPEECH*, pages 2769–2773.

Gideon Mendels, Erica Cooper, Victor Soto, Julia Hirschberg, M.J.F. Gales, Kate Knill, Anton Ragni, and Haipeng Wang. 2015. Improving speech recognition and keyword search for low resource languages using web data. In *Proc. INTERSPEECH*.

Raghav Menon, Herman Kamper, John Quinn, and Thomas Niesler. 2018a. Fast asr-free and almost zero-resource keyword spotting using dtw and cnns for humanitarian monitoring. In *Proc. INTERSPEECH*, pages 2608–2612.

Raghav Menon, Herman Kamper, Emre Yılmaz, John Quinn, and Thomas Niesler. 2018b. Asr-free cnn-dtw keyword spotting using multilingual bottleneck features for almost zero-resource languages. In *Proc. SLTU*, pages 20–24.

Vikramjit Mitra, Julien van Hout, Horacio Franco, Dimitra Vergyri, Yun Lei, Martin Graciarena, Yik-Cheung Tam, and Jing Zheng. 2014. Feature fusion for high-accuracy keyword spotting. In *Proc. ICASSP*, pages 7143–7147.

Anna Piunova, Eugen Beck, Ralf Schlüter, and Hermann Ney. 2019. Rescoring keyword search confidence estimates with graph-based re-ranking using acoustic word embeddings. In *Proc. INTERSPEECH*, pages 4205–4209.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

A. Ragni, C. Wu, M. J. F. Gales, J. Vasilakes, and K. M. Knill. 2017. Stimulated training for automatic speech recognition and keyword search in limited resource conditions. In *Proc. ICASSP*, pages 4830–4834.

Shakti Rath, Kate Knill, A. Ragni, and M.J.F. Gales. 2014. Combining tandem and hybrid systems for improved speech recognition and keyword spotting on low resource languages. In *Proc. INTERSPEECH*, pages 835–839.

Korbinian Riedhammer, Van Hai Do, and James Hieronymus. 2013. A study on lvcsr and keyword search for tagalog. In *Proc. INTERSPEECH*.

Jan Robin Rohlicek, William Russell, Salim Roukos, and Herbert Gish. 1989. Continuous hidden markov modeling for speaker-independent word spotting. In *Proc. ICASSP*, volume 1, pages 627–630.

Richard C. Rose and Douglas B. Paul. 1990. A hidden markov model based keyword recognition system. In *Proc. ICASSP*, volume 1, pages 129–132.

Tara Sainath and Carolina Parada. 2015. Convolutional neural networks for small-footprint keyword spotting. In *Proc. INTERSPEECH*, pages 1478–1482.

Ming Sun, David Snyder, Yixin Gao, Varun K. Nagaraja, Mike Rodehorst, Sankaran Panchapagesan, Nikko Strom, Spyridon Matsoukas, and Shiv Naga Prasad Vitaladevuni. 2017. Compressed time delay neural network for small-footprint keyword spotting. In *Proc. INTERSPEECH*, pages 3607–3611.

Igor Szöke, Petr Schwarz, Pavel Matejka, Lukás Burget, Martin Karafiát, and Jan Honza Cernocký. 2005. Phoneme based acoustics keyword spotting in informal continuous speech. In *TSD*.

K. Thambiratnam and S. Sridharan. 2005. Dynamic match phone-lattice searches for very fast and accurate unrestricted vocabulary keyword spotting. In *Proc. ICASSP*, volume 1, pages I/465–I/468.

Voxforge.org. Free speech... recognition (linux, windows and mac) - voxforge.org. http://www.voxforge.org/. Accessed 06/25/2014.

Jay G. Wilpon, Lawrence R. Rabiner, Chin-Hui Lee, and E. R. Goldman. 1990. Automatic recognition of keywords in unconstrained speech using hidden markov models. *IEEE Trans. Acoust. Speech Signal Process.*, 38:1870–1878.

Martin Wollmer, Florian Eyben, Joseph Keshet, Alex Graves, Bjorn Schuller, and Gerhard Rigoll. 2009. Robust discriminative keyword spotting for emotionally colored spontaneous speech using bidirectional lstm networks. In *Proc. ICASSP*, pages 3949–3952.

Steve Young, M.G. Brown, J.T. Foote, Gareth Jones, and K. Jones. 1997. Acoustic indexing for multimedia retrieval and browsing. In *Proc. ICASSP*, volume 1, pages 199 – 202.

Hierarchical Multi-Task Transformers for Crosslingual Low Resource Phoneme Recognition

Kevin Glocker[1], **Munir Georges**[1,2]
[1]Technische Hochschule Ingolstadt, Research Institute AImotion Bavaria, Ingolstadt, Germany
[2]Intel Labs Germany
kevin.glocker@thi.de, munir.georges@thi.de

Abstract

This paper proposes a method for multilingual phoneme recognition in unseen, low resource languages. We propose a novel hierarchical multi-task classifier built on a hybrid convolution-transformer acoustic architecture where articulatory attribute and phoneme classifiers are optimized jointly.

The model was evaluated on a subset of 24 languages from the Mozilla Common Voice corpus. We found that when using regular multi-task learning, negative transfer effects occurred between attribute and phoneme classifiers. They were reduced by the hierarchical architecture. When evaluating zero-shot crosslingual transfer on a data set with 95 languages, our hierarchical multi-task classifier achieves an absolute PER improvement of 2.78% compared to a phoneme-only baseline.

1 Introduction

While many highly effective architectures for speech recognition have been introduced in recent years, most require large amounts of language-specific training data. However, for a substantial portion of the worlds languages, only few or no annotated speech recordings are available for training or fine-tuning. To leverage the accuracy of end-to-end architectures, systems intended for low-resource ASR are often (pre-)trained on large multilingual corpora from mostly high-resource languages such as in Xu et al. (2021), who fine-tune a multilingually pretrained wav2vec 2.0 model for the crosslingual transfer task. They are either fine-tuned on low resource languages as evaluated by, e.g., Siminyu et al. (2021) or directly applied zero-shot, as outlined by Li et al. (2021a).

Several systems have been introduced that use articulatory attribute systems developed by linguists to improve phoneme recognition performance. In such systems, attributes are primarily used as an input in the form of trainable embeddings for each attribute individually as proposed by, e.g., Li et al. (2021a) or for feature vectors as in, e.g., Zhu et al. (2021), or using signature matrices as described by, e.g, Li et al. (2020). In contrast, Lee et al. (2019) applied multi-task learning to train separate articulatory feature classifiers and triphone states using shared layers for Mandarin at the same time with a TDNN architecture on forced alignments.

In this work, a multilingual phoneme recognition architecture is introduced. It is derived from a similar architecture applied to computer assisted pronunciation training in Mandarin (Glocker, 2021). Hierarchical multi-task learning is used to learn jointly to classify articulatory attributes and phonemes with an additional direct connection between the attribute and the phoneme classifier.

The proposed acoustic model for phoneme recognition is introduced in Section 2. The system is then evaluated in Section 3 in the high resource and zero-shot crosslingual settings. Afterwards, results are discussed and the paper concluded in Section 4.

2 Crosslingual Phoneme Recognition

Section 2.1 describes the hybrid transformer-acoustic model for encoding frame sequence. The hierarchical multi-task classifier for articulatory attributes and phonemes is introduced in Section 2.2.

2.1 Transformer Acoustic Model

A hybrid convolution and transformer encoder model is used for acoustic sequence modeling as shown in Figure 1. The architecture and hyperparameter choices are derived from the transformer model introduced by Synnaeve et al. (2019). First, the audio is resampled to 16kHz. 40 dimensional MFCC features using 25ms frames with a stride of 10ms are extracted. The features are then passed into two GLU-activated convolution layers to encode local context, with a kernel size of three and 512 and 400 channels respectively. Each convolution layer is preceded by layer normalization and

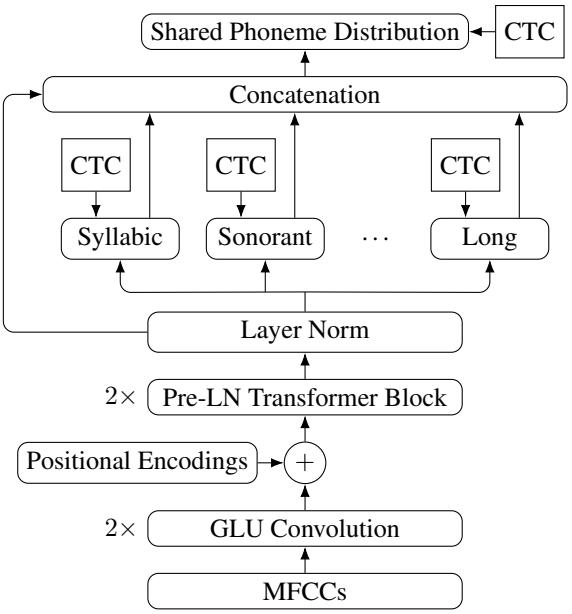


Figure 1: Illustration of the hybrid convolutional transformer phoneme recognition model with the hierarchical connections between attribute and phoneme classifiers

followed by a dropout layer for regularization. A stride of 2 is used in the second GLU layer, increasing the receptive field of the model to 5 frames while keeping the output lengths shorter than the length of phoneme sequences for CTC.

Sinusoidal positional encodings as proposed by Vaswani et al. (2017) are added to the output representations of the convolution layers. The sequence is passed through a shallow 2-layer transformer. In the transformer, Pre-LN transformer blocks are used without warmup as proposed by Xiong et al. (2020). Feedforward layers with a hidden size of 2048 and 4 attention heads are used motivated by Vaswani et al. (2017). The dropout rate is 0.2.

2.2 Hierarchical Multi-Task Classifiers

In contrast to previous work (Lee et al., 2019), classifiers are not trained completely independently but are connected in a hierarchical structure. Cascading information between tasks has also previously been successfully applied to jointly optimizing NLP tasks at different "levels" such as POS and dependency parsing (Crawshaw, 2020).

In the hierarchy, both the attribute and phoneme classifiers take the normalized output of the transformer acoustic model as an input. In addition to the acoustic representation, the phoneme classifier receives a concatenation of the probability distributions from each articulatory attribute classifier. More specifically, for each time step t given a set of attribute classifier logits A_t, the transformer hidden vector h_t, and the weights and biases of the phoneme projection layer W and b, the phoneme logits p_t are computed as follows:

$$v_t = \left(\bigoplus_{a \in A_t} \text{softmax}(a)\right) \oplus h_t \quad (1)$$

$$p_t = W^T v_t + b \quad (2)$$

Each classification layer is then independently but simultaneously optimized using connectionist temporal classification (CTC; Graves et al. (2006)). For consistency, articulatory attribute vectors are directly mapped to each phoneme without merging repetitions. As a result, there is always a 1:1 correspondence between attribute and feature labels at training time. While the attribute and phoneme classifiers form a flat hierarchy in this work, the hierarchical structure generalizes to any directed acyclic graph representing phonetic feature structures.

3 Evaluation

We evaluated the proposed hierarchical multi-task transformer with two experiments.

(1) In the "Multi-Task" variant, regular multi-task learning is used where attribute probabilities are not used as inputs to the phoneme classifier.

(2) In the "Phonemes Only" model, only the phoneme classifier is used and attribute information is only applied to phoneme mapping at test time.

Batch sizes are set dynamically for efficiency until the product of the batch and frame sequence dimensions reaches 320,000. The Adam optimizer (Kingma and Ba, 2015) was used for training with $\beta_1 = 0.9$ and $\beta_2 = 0.98$ as in Vaswani et al. (2017). A learning rate of 0.001 is used. The training was stopped once the average validation set losses did not decrease for more than 3 epochs.

The transformer acoustic model was implemented in the PyTorch framework (Paszke et al., 2019), using Torchaudio (Yang et al., 2021) for Audio processing and feature extraction.

The data sets for training and evaluation are described in Section 3.1. Section 3.2 presents and analyses the results for phoneme and attribute classification for high and low resource languages.

3.1 Datasets

For training and evaluation in the high resource setting, version 10.0 of the Mozilla Common Voice corpus was used, which contains crowdsourced recordings of sentences. Each sentence is tokenized

using Stanza (Qi et al., 2020), after which punctuation is removed and each token is transcribed into phonemes using Epitran (Mortensen et al., 2018). Finally, the transcriptions are segmented according to the IPA segments available in the Panphon database (Mortensen et al., 2016) for the phoneme inventory extracted from the training data for each language. The 24 articulatory attributes from Panphon are used for creating and supervising the attribute classifiers. The multilingual training set was constructed from at most 15,000 sentences from the training sets of 24 languages from Common Voice, for which both a tokenization and a grapheme-to-phoneme model is available. The original development and test sets were used unchanged.

The first release[1] of the multilingual corpus published by Li et al. (2021b) is used for evaluating zero-shot transfer in this work as in Li et al. (2021a). It provides 5,509 validated utterances with phoneme transcriptions for 95 low-resources languages from five continents. Since recordings for Czech, Dutch, Maltese, Hindi and Hungarian are also included in the training data, they are removed from the test data before computing the averages.

To handle different inventories and OOV phonemes in the test languages, phonemes predicted by the model are mapped to each target inventory using the hamming distance between attribute vectors. This corresponds to the "tr2tgt" approach introduced by Xu et al. (2021). For the UCLA Phonetic Corpus, the included inventory files are used for this mapping even if they include a phoneme that doesn't appear in a transcription.

3.2 Experiments

The overall performance on phoneme and articulatory attribute detection on Common Voice can be seen in Table 1. In addition to the phoneme error rate (PER), the attribute error rate (AER) is computed for each attribute individually and then averaged over all attributes. The hierarchical multi-task model reaches lower PER and average AER than regular multi-task learning in both the high and low resource setting. The regular multi-task model also performs worse than the phoneme only baseline. This shows, that negative transfer effects are stronger without the hierarchical connection.

Compared to the "Phonemes Only" model, the hierarchical model performs almost identically in the high-resource setting. However, as shown in Table 2, there is an improvement to the unseen low-resource languages from the UCLA Phonetic Corpus. In contrast, the regular multi-task model also yields higher PERs in this setting.

Architecture	%PER	%AER
Phonemes Only	48.96	–
Multi-Task	52.19	19.43
Hierarchical Multi-Task	49.11	17.99

Table 1: Average phoneme and attribute error rates for the Common Voice subset representing the high resource setting

Architecture	%PER	%AER
Phonemes Only	74.77	–
Multi-Task	75.28	34.14
Hierarchical Multi-Task	71.99	30.25

Table 2: Average phoneme and attribute error rates for the UCLA Phonetic Corpus representing the low resource setting

Figure 2 shows the phoneme and average attribute error rates for the Common Voice test sets of the languages used for training. The variance of PERs between languages is high ($\sigma^2 = 135.03$). On the attribute level, the variance of the AER between languages is much less pronounced ($\sigma^2 = 15.61$) and lower AER doesn't correlate with higher PER ($r^2 = 0.016$). For instance, the PER is highest for Arabic and Vietnamese even though their AER are among the lowest in the test set.

Since the AER was improved most consistently across languages through the hierarchical architecture, research into better modeling the connection between articulatory attributes and phonemes could lead to larger PER improvements in future work.

For Arabic and Urdu, a contributing factor might be Epitran not transcribing short vowels since they are not present in their orthography (Mortensen et al., 2018). For Vietnamese, the higher PER is likely due to it being the second-lowest resource language in the training data with only 2259 validated utterances and one of only two tonal languages alongside Thai.

In contrast, phoneme recognition is the most accurate for the five romance languages including Spanish, Italian and Catalan. They likely benefit the most from the multilingual settings since they

[1] https://github.com/xinjli/ucla-phonetic-corpus/releases/tag/v1.0

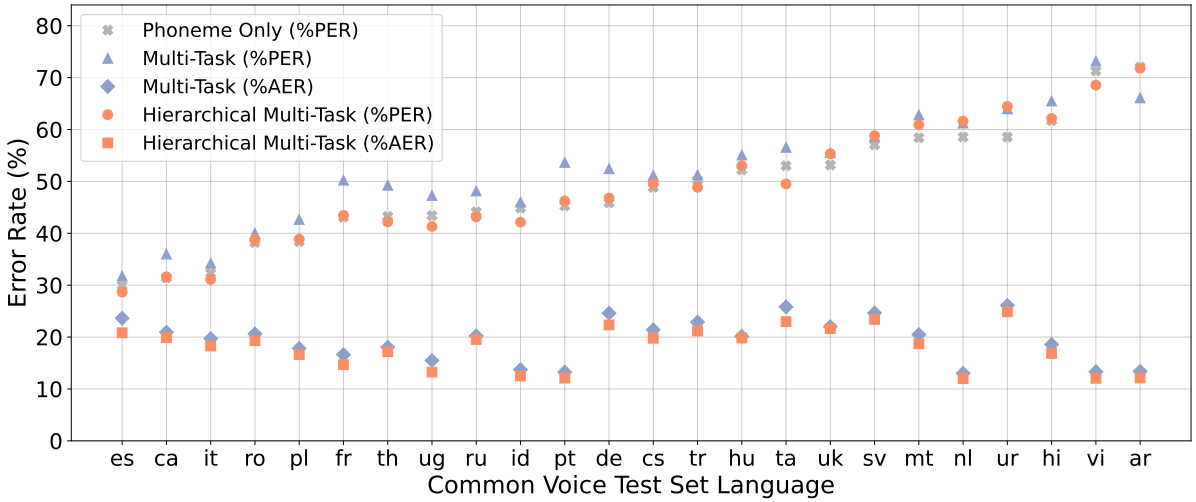


Figure 2: Phoneme Error Rates (PER) and the averages over all Attribute Error Rates (AER) on the test sets from Common Voice for the languages used for training

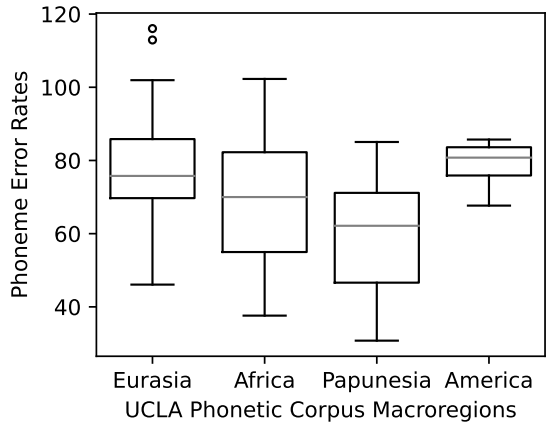


Figure 3: Phoneme Error Rates (PER) for the languages in the UCLA Phonetic Corpus grouped into macroregions accordig to Glottolog (Hammarström et al., 2022)

are closely related.

A possible explanation for the low correlation between AER and PER is, that the frame level probabilities tend to form single frame spikes when trained with CTC (Graves et al., 2006). Since CTC loss is computed for every classifier independently, spikes for attributes of the same phoneme sometimes occur on different frames. As a result, the phoneme classifier is likely to receive high blank probabilities from multiple attribute classifiers.

The crosslingual transfer results are further divided into macroregions in Figure 3 based on Glottolog (Hammarström et al., 2022). The model transfers best to the set of 10 languages from the "Papunesia" region, despite there being no languages from this region in the training set. In contrast, the model generalizes poorly to the four American languages. Some outliers with particularly high PER might also be caused by the noisy conditions under which some utterances were recorded (Li et al., 2021b).

4 Conclusion

A novel hierarchical multi-task architecture is presented and evaluated together with a hybrid convolution-transformer acoustic model for phoneme classification. In contrast to regular multi-task learning, the phoneme classifier receives attribute probabilities as additional inputs.

It tackles the crosslingual transfer task for phoneme recognition in low resource languages. For zero-shot classification in such languages, only their phoneme inventory is required.

Negative transfer effects observed in regular multi-task learning were reduced. When evaluated on the UCLA Phonetic Corpus, the proposed system yielded an absolute phoneme error rate reduction of 2.78% across 95 unseen languages compared to a phoneme-only baseline.

Future work may investigate the low correlation between AER and PER, and further analyse the cause of the high variance of PER between languages. In particular, we plan to investigate and improve the mapping between the shared phoneme inventory and language specific inventories to tackle these challenges. Furthermore, tones could be moved to their own layer in the hierarchy to better reflect their suprasegmental nature.

References

Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *CoRR*, abs/2009.09796.

Kevin Glocker. 2021. Unsupervised end-to-end computer-assisted pronunciation training for mandarin. Master's thesis, Eberhard Karls Universität Tübingen.

Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. *Proceedings of the 23rd international conference on Machine learning*.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank. 2022. glottolog/glottolog: Glottolog database 4.6.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Yueh-Ting Lee, Xuan-Bo Chen, Hung-Shin Lee, Jyh-Shing Roger Jang, and Hsin-Min Wang. 2019. Multi-task learning for acoustic modeling using articulatory attributes. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 855–861.

Xinjian Li, Siddharth Dalmia, David R. Mortensen, Juncheng Li, Alan W. Black, and Florian Metze. 2020. Towards zero-shot learning for automatic phonemic transcription. In *AAAI*.

Xinjian Li, Juncheng Li, Florian Metze, and Alan W. Black. 2021a. Hierarchical Phone Recognition with Compositional Phonetics. In *Proc. Interspeech 2021*, pages 2461–2465.

Xinjian Li, David R. Mortensen, Florian Metze, and Alan W Black. 2021b. Multilingual phonetic dataset for low resource speech recognition. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6958–6962.

David R. Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision G2P for many languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

David R. Mortensen, Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer, and Lori S. Levin. 2016. Panphon: A resource for mapping IPA segments to articulatory feature vectors. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3475–3484. ACL.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

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*.

Kathleen Siminyu, Xinjian Li, Antonios Anastasopoulos, David R. Mortensen, Michael R. Marlo, and Graham Neubig. 2021. Phoneme Recognition Through Fine Tuning of Phonetic Representations: A Case Study on Luhya Language Varieties. In *Proc. Interspeech 2021*, pages 271–275.

Gabriel Synnaeve, Qiantong Xu, Jacob Kahn, Edouard Grave, Tatiana Likhomanenko, Vineel Pratap, Anuroop Sriram, Vitaliy Liptchinsky, and Ronan Collobert. 2019. End-to-end asr: from supervised to semi-supervised learning with modern architectures. *ArXiv*, abs/1911.08460.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR.

Qiantong Xu, Alexei Baevski, and Michael Auli. 2021. Simple and effective zero-shot cross-lingual phoneme recognition. *ArXiv*, abs/2109.11680.

Yao-Yuan Yang, Moto Hira, Zhaoheng Ni, Anjali Chourdia, Artyom Astafurov, Caroline Chen, Ching-Feng Yeh, Christian Puhrsch, David Pollack, Dmitriy Genzel, Donny Greenberg, Edward Z. Yang, Jason Lian, Jay Mahadeokar, Jeff Hwang, Ji Chen, Peter Goldsborough, Prabhat Roy, Sean Narenthiran, Shinji Watanabe, Soumith Chintala, Vincent Quenneville-Bélair, and Yangyang Shi. 2021. Torchaudio: Building blocks for audio and speech processing. *arXiv preprint arXiv:2110.15018*.

Chengrui Zhu, Keyu An, Huahuan Zheng, and Zhijian Ou. 2021. Multilingual and crosslingual speech

recognition using phonological-vector based phone embeddings. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1034–1041.

Concatenative Phonetic Synthesis for the Proto-Indo-European Language

Patrick J. Donnelly
Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR, USA
patrick.donnelly@oregonstate.edu

Abstract

We propose a flexible concatenative text-to-speech system to synthesize hypothesized pronunciations of the reconstructed Proto-Indo-European language. To accomplish this, we synthesize speech examples in 100 extant languages and extract individual phones. Using this large database of phonetic sounds, we concatenate individual phonemes together to estimate pronunciations of Proto-Indo-European. Where available, we prioritize consecutive phones from the same source to help increase naturalness and intelligibility of the synthesized speech. Since the language's precise pronunciation is debated, we provide an interface to select the specific phonetic symbol(s) used for each of the language's phonemes and diphthongs. We provide this novel interactive tool to enable researchers and students to aurally explore the different and competing phonological hypotheses debated in the literature.

1 Introduction

Proto-Indo-European (PIE) is the reconstructed ancestor of all Indo-European languages. PIE is hypothesized to have been spoken as a single language sometime during the late Neolithic through the Early Bronze Age (between 4500 to 2500 BCE). According to the Kurgan hypothesis (Gimbutas, 1956), the language likely originated in the Pontic-Caspian steepe of eastern Europe. Over the following centuries, waves of Indo-European (IE) peoples migrated across much of the Eurasian continent. As they dispersed, their language split and underwent shifts in pronunciation, changes in morphology, and acquisitions of new vocabulary. This process continued for centuries, resulting in 448 extant daughter languages across eight subfamilies.[1]

[1] https://www.ethnologue.com/subgroups/indo-european

There is no historical record of PIE. Like other proto languages, the language was meticulously reconstructed using the comparative method (Hoenigswald, 1963). Although Indo-European (IE) linguists have largely converged on the phonemic inventory of PIE, there remains ongoing debate about the interpretation of these phonemes. Unlike living languages, there does not exist an agreed upon mapping of phonemes in PIE to specific phonetic symbols in the International Phonetic Alphabet (IPA). For this reason, there have not been previous attempts to synthesize PIE speech.

In this work we present a text-to-speech system that attempts to estimate PIE speech given a specific mapping of phonemes to IPA pronunciations. We sample phonetic sounds from 100 modern languages to build a concatenative speech synthesizer which is able to pronounce text in the reconstructed Proto-Indo-European language. We provide a flexible approach that allows listeners to tune the phonology to enable aural realizations of different hypothetical pronunciations. To our knowledge, this is the first attempt at speech synthesis for a prehistoric reconstructed language.

2 Proto-Indo-European Language

The phonology of PIE has been reconstructed based on the phonology of extant IE languages. This scholarship initially relied upon modern and well attested historical languages, such as Latin, Ancient Greek, and Vedic Sanskrit. However, the surprise discoveries of the Hittite and Tocharian languages in the early 20th Century provided new scholarly evidence that led to new understandings and sparked new academic debates (Jasanoff, 2017).

Hundreds of words have been reconstructed in PIE and scholars have largely converged on the morphology, although areas of debate still remain. In 1868, the linguist August Schleicher com-

		Labial	Coronal	Dorsal			Laryngeal
				palatal	plain	labial	
Nasals		*m	*n				
	voiceless	*p	*t	*ḱ	*k	*k^w	
Stops	voiced	(*b)	*d	*ǵ	*g	*g^w	
	aspirated	*b^h	*d^h	*$ǵ^h$	*g^h	*g^{wh}	
Fricatives			*s				*h_1, *h_2, *h_3
Liquids			*r, *l				
Semivowels				*y		*w	

Table 1: Common notation used for Proto-Indo-European phonology (Kapović, 2017). The preceding asterisk (*) denotes the phoneme is reconstructed rather than attested. The symbol *b is disputed and shown in parenthesis. The superscripts ^{h} and ^{w} stands for aspiration and labialization, respectively. The symbols *h_1, *h_2, *h_3 serve as phonemes for the three unknown laryngeal sounds. The phoneme *y represents the palatial semivowel (IPA /j/).

posed the short story "*The Sheep and the Horses*" ("H_2ówis h_1éḱwōs-k^we") in his version of reconstructed PIE (Adams, 1997). Over the decades, linguistics have published revisions to this story, accounting for new consensuses in the field or to advance their own linguistic hypotheses. In the absence of any text in PIE, this story has come to serve as the standard mechanism to demonstrate and compare different reconstructions. More recently several Indo-Europeanists linguists collaborated to each reconstruct their versions of another short story entitled "*The King and the god*" ("H_3réḱs deywós-k^we") (Adams, 1997).

2.1 PIE Phonology

Although linguistics have generally converged on the phonetic inventory of PIE, there remains significant debate regarding the pronunciation of these phonemes (Kapović, 2017). The pronunciations of certain sounds in PIE are not known, and may never been known. The majority of phonetic controversy concerns two issues. The first debate pertains to the pronunciation of the series of plosive stops. The second debate pertains to the belief that PIE had a set of phonemes that are not attested to in any extant daughter language.

2.1.1 Glottalic Theory

Glottalic theory proposes that PIE had ejective stops (*p', *t', *k') instead of the traditionally reconstructed plain voiced stops (*b, *d, *g). Once popular, this theory is no longer widely accepted by historical linguists (Barrack, 2002). The reconstruction of these phonemes is made more difficult by the *centum-satem* language split that divides northern and southern IE languages. This divide is named for the pronunciation of the word "hundred" in early PIE languages (Greek vs. Sanskrit). In centum languages, the plain and palatovelears merged together, while in the satem languages, the plain and labiovelars merged together.

2.1.2 Laryngeal Theory

The other controversy surrounding PIE phonology pertains to the number and pronunciation of vowels in PIE. Laryngeal theory proposes that there existed at least three sounds which do not survive in any extant daughter languages. Once reconstructed with five vowels, PIE is now commonly reconstructed with only two vowels, [e] and [o], that were colored by three hypothesized laryngeal sounds: *h_1, *h_2, and *h_3. Today most linguists accept the existence of the laryngeals but continue to dispute their exact phonetic realization (Keiler, 2015). As such, there exist numerous competing interpretations and revisions in the scholarly literature.

2.2 Open Questions

Given these significant open questions regarding its pronunciation, there have not been any previous attempts at automatic speech synthesis of PIE. Most modern synthesis approaches that seek to produce naturalistic speech require extensive knowledge about the pronunciation rules of the language or large datasets of spoken examples. Therefore, speech synthesis using cutting edge technologies is not readily possible for reconstructed languages.

3 Speech Synthesis

Speech synthesis is the task of converting written text into an audio waveform that represents a machine generated realization of the spoken text. These systems are also known as Text-to-Speech (TTS) tools. The majority of approaches for speech

synthesis utilize large corpora of text and audio examples. Accordingly, the majority of research in speech synthesis has focused on widely-spoken languages, particularly those with global influence.

Despite calls for more speech recognition tools for under-resourced languages (Besacier et al., 2014), there has been relatively little work in TTS for most of the world's languages. Nevertheless, speech synthesizers have been built for historical languages, such as Latin and Greek; constructed languages, such as Esperanto and Klingon (Jokisch and Eichner, 2000); and some endangered European languages like Basque or Irish (Chasaide et al., 2017). In an encouraging direction, the authors of a recent study collected and analyzed corpora, documented phonology, and built TTS systems for 12 different African languages (Ogayo et al., 2022).

Researchers have been attempting speech synthesis since the 1950's. Here we briefly review the principle categories of speech synthesis models.

3.1 Articulation Synthesis

Articulation Synthesis is a physical model which emulates various aspects of human pronunciation to synthesize speech. They provide tuneable parameters for the various aspects of human pronunciations (e.g., tongue, pharynx, vocal chords, etc.). While articulation synthesizers are able to produce high-quality speech, the large number of parameters make these systems computationally expensive. These constraints limit their practicality in real-time deployment. Neil Thapen's interactive Pink Trombone[2] is a recent interactive example of articulation synthesis.

3.2 Formant Synthesis

Formant Synthesis generates speech by sending a source signal through a series of filters modeling different formants of the desired speech sound. Speech created with formant synthesis often sounds less naturalistic than other approaches, but it is fast to produce and is generally intelligible (Lukose and Upadhya, 2017). In this work, we make use of the popular formant synthesizer `eSpeak-ng`[3] to generate various phonetic sounds across different languages spoken by a single artificial speaker.

[2] https://imaginary.github.io/pink-trombone/

[3] https://github.com/espeak-ng/espeak-ng

3.3 Concatenative Speech Synthesis

Concatenative Speech Synthesis is another traditional approach that relies upon large databases of sounds vocalized by the same speaker. These systems concatenate different prerecorded waveforms together in order to vocalize the desired text. Depending on the system, the individual constituent waveforms may be phones or phonemes, syllables, or entire words. Speech produced with concatenative synthesis is often intelligible but can potentially sound rather unnatural. This occurs because of the limited syntactic and semantic context as different sounds are spliced together at a very low structural level (Khan and Chitode, 2016).

3.4 Machine Learning Approaches

In recent years, these aforementioned signal processing methods have largely been superseded by new approaches using machine learning. Because these new approaches require large corpora of spoken audio examples, they are not suited for our attempt to estimate pronunciation of a non-attested reconstructed language. These approaches are beyond the scope of this work, but we briefly review the important recent developments here.

Techniques using statistical parametric estimation, such as hidden Markov models, consistently achieved robust and intelligible speech synthesis (Yamagishi et al., 2009). However, the recent abundance of big data and advances in deep learning algorithms have led to new speech synthesis techniques that dominant use in modern day TTS applications and research directions (see review (Ning et al., 2019)).

One such example is Wavenet, an autoregressive generative model for end-to-end speech synthesis (Oord et al., 2016) which models the waveform directly, without requiring a hybrid model or other processes to assemble the synthesized speech. Another such production-quality model, Deep Voice, synthesizes speech entirely from deep neural networks (Arık et al., 2017). Ongoing research in speech synthesis continues in many areas, such as emotional, dialogic, and spontaneous speech production (Delić et al., 2019).

3.5 Our Approach

Our approach requires the use of sets of phonetic sounds that do not occur together in any single language. And we require these sounds to originate from the same speaker. Unfortunately, there has

been very little development into multi-language speech synthesis (Malcangi and Grew, 2010). None of the aforementioned modeling techniques are particularly well suited for this task alone. For similar reasons, concatenative synthesis has previously been used to study some under-resourced languages (Van Niekerk and Barnard, 2009).

In this work we use formant synthesis to produce a library of spoken sounds in various languages. We then use concatenative synthesis to build speech from this corpus of sampled sounds.

4 Database of Phones

We present an approach that attempts to estimate the speech in PIE by concatenating different phonetic sounds extracted from various languages. To accomplish this task, we require phonemes across many different languages spoken by the same speaker. This is necessary in order to maintain a continuity of acoustic characteristics across the concatenation points. To build this dataset, we generate word lists in multiple languages, synthesize speech utterances for each word, splice the audio by individual phoneme, and save these phonetic sounds to a database.

4.1 Sampling Languages

To generate multi-language speech using the same speaker, we use the open-source `eSpeak-ng` tool, a popular TTS engine. `eSpeak-ng` currently supports 127 languages and accents.

For each language available in `eSpeak-ng`, users have carefully crafted lists of phonemes, pronunciation rules, and example words. These specific pronunciation rules allow the synthesizer to generate more realistic speech by considering the pronunciation context of phonemes, syllables, and words. These rules are developed with feedback from native speakers, and subsequently the languages available in `eSpeak-ng` reflect only those languages with a very large number of speakers.

Of the 127 available languages, we exclude constructed languages such as Esperanto or Klingon but we include the IE languages of Ancient Greek and Latin. For English, we retained only standard American and British pronunciation, and we exclude other less common dialects. For Spanish and Portuguese, we include both European and Latin American pronunciation. Although the majority of remaining languages are IE languages (56%), we also include non-IE languages in order to increase our phonetic inventory. Among those, we included three dialects of Chinese: Cantonese, Hakka and Mandarin. Altogether, we select 100 unique languages and five additional dialects for a total of 105 speakers from whom we synthesize speech.

4.2 Swadesh Lists

Swadesh lists were devised by linguist Morris Swadesh as a tool when measuring relationships between languages using glottochronology. A Swadesh list contains 207 words in a particular language (Swadesh, 1952). Given their long use in comparative linguistics, there exist complete or partial lists for many of the world's languages. We collect these lists for our 100 selected languages from Wiktionary.[4]

Some of these lists contain multiple synonyms or variants for each word. We exclude stand-alone suffices, but otherwise accept all complete words. Our goal is not to compare phonology but to generate a sampling of many possible phonemic pronunciations in the language. Therefore the number of words synthesized varies between languages.

4.3 Generating Phonemes

Next we synthesized each of these words using Praat.[5] Praat is popular tool for speech analysis in phonetics that also provides speech synthesis using `eSpeak-ng`. When generating speech, Praat labels and segments each of the phonemes used to create the synthesized utterance, using the Kirshenbaum phonetic encoding notation for IPA. This provides very specific IPA notation for each individual phoneme. These include various articulation diacritics, co-articulations, and diphthongs.

Praat supports a scripting language, which we use to automate the following tasks. For each language we generate a speech synthesizer. We use the default speaker voice "Female 1" and a speech rate of 150 words per minute. For each word in the language word list, we automatically generate the synthesized utterance. We iterate across the waveform to split and save each excerpt representing a single phoneme. We save these phonetic sounds as a single channel 16k Hz `wav` file. We snip all audio at zero-crossings in the waveform to prevent sonic artifacts when concatenating sounds together. We henceforth refer to these excerpts as phones, indicating that they have been extracted from the

[4] https://en.wiktionary.org/wiki/Appendix:Swadesh_lists

[5] https://www.praat.org

phonemic context of their source languages. Later, we will use sets of these phones to approximate phonemes in PIE. In a database, we log the source for each phone, and we make note of its context by logging the phones that precede and follow it.

4.4 Phonetic Inventory

In total, we collected 124,252 audio samples covering 339 uniquely labeled phonetic sounds. This set includes many short and long vowels, diphthongs, and consonantal articulations. Because the generated phones are shaped by the phonemic context within the source word and the specific pronunciation rules of their language, the duration of the excerpts differ even among sounds with the same precise phonetic IPA annotation.

Some of the sounds hypothesized to exist in PIE no longer occur in any IE daughter languages. While they do survive in some of the world's extant languages, they are quite rare. One such example is the voiced uvular plosive /ɢ/ (Prescott, 2018) which does not occur in the languages available.

Among our phonetic sounds extracted from these 100 widely spoken languages, we find examples of aspiration (e.g., /g^h/) and palatalization (e.g., /g^j/) but not examples of labialization (e.g., /g^w/). Instead we will need to approximate a few hypothesized sounds (e.g., /k^w/, /g^w/, and /g^{wh}/) using pairs of phones. For example, we substitute the sequence of two consecutive phones /gw/ for /g^w/. In languages such as Welsh, this sound occurs frequently, descended from this origin in PIE.

5 Concatenative Synthesizer

In this section we outline our concatenative speech synthesizer and describe our user interface.

5.1 Text Processing

In order to prepare the user-provided text for phonetic matching against our database of phones, we must perform a number of text manipulation steps. First, we normalize whitespace and convert the text to lowercase. Next, we split the text into individual sentences and phrases. We then split these phrases further into individual words. Lastly, we remap alternate orthography to a common notation. For example, the graphemes *ĝ and *ĝ are both remapped to *ǵ. Finally, we tokenize each word into phonemes in PIE, greedily grouping characters together to match the phonological notation given in Table 1.

5.2 Mapping Phonemes

In this step, we map the different PIE phonemes to specific pronunciations in IPA. Because there is no established pronunciation for PIE, our goal is to provide a flexible tool to realize different hypothetical pronunciations. To accomplish this, we read a JSON file containing a mapping of PIE phonemes to phonetic sounds. This mapping can be specified by the user at run-time to guide the desired pronunciation of the synthesized speech. The user is able to assign an IPA symbol to each consonant, vowel, and vowel-semivowel diphthong (e.g., *ew, *oy).

We also provide the user control over the pronunciation rules for the vowels following each of the unknown laryngeals *h_1, *h_2, *h_3. The potential "coloring" of the vowels following these sounds is an important part of hypotheses subscribing to Laryngeal Theory. Evidenced by pronunciations across its descendants, the presence of *h_2 is thought to color the vowel *e to *a while *h_3 colors *e to *o. Additionally, when the laryngeal occurs after the vowel, it likely lengthens the vowel.

5.3 Matching Phones

For each PIE phoneme in a word, we search our database for an example that represents the target sound. Of those found, we then examine the phone that precedes and that follows our target phone. We first attempt to retrieve a tri-gram of consecutive phones taken from a single source word. When unavailable, we next prioritize retrieving a bi-gram of phones. Finally, when such a pair is unavailable, we resort to a single phone. Our approach prioritizes finding consecutive phones from a single source word in order to attempt a more natural pronunciation. This is especially beneficial in the cases of short syllables or voiceless consonants whose pronunciation is shaped by adjacent vowels.

In this task, we consider silence as a possible target sound, which allows us to explicitly find sounds that start or end a syllable when present in the data. We prioritize matching these phones at the beginning and end of syllables to help shape a more natural pronunciation. In particular, this process helps reflect the natural attack and release that occurs at the beginning and end of words.

5.4 Audio Manipulation

Next, we manipulate the audio snippets that correspond to accented vowels. We provide three possibilities for handling the accent. In the first, we

ignore the accent altogether and use the unaltered phone. In the second option, we provide a stress accent. To simulate a stressed pronunciation, we increase the amplitude of the waveform containing the vowel by +3 dB. In the third option, we apply a pitch accent. To simulate a pitch accent, we raise the frequencies of the waveform by one-tenth of an octave. These default threshold values were selected to sound reasonable but can readily be tuned.

Once matching audio files have been identified for each PIE phoneme in the input, we concatenate these phones together. We add pauses of silence between syllables, words, and sentences. To help make the speech sound slightly more natural, we randomize the amount of silence added, scaled by the type of the pause. As a final step, we export a single channel 16 kHz audio file that can be saved to disk or displayed on an interactive web-page.

5.5 User Interface

We provide a graphical interface to demonstrate our concatenative speech synthesis tool, built in Python using the data science interface Streamlit.io.[6] This interface, shown in Figure 1, allows a user to select a particular mapping from each PIE phoneme to a specific IPA pronunciation. For each phoneme, we provide various possibilities hypothesized in the literature (Swiggers, 1989; Beekes, 2011; Meier-Brügger, 2013; Kapović, 2017; Byrd, 2018). However, some of these options are limited by what is available in our dataset of phonetic sounds.

After selecting the phonology, the user can enter a PIE word or phrase in the textbox. After the user clicks the "Speak" button, the app syntheses the text to speech using the process outlined in Section 5. This process is quick but may take a few seconds for longer texts. The interface then adds an audio player widget, allowing the user to listen to the speech or save the file to disk. Below the audio player, the interface prints the IPA transcription that was used to produce the speech. The user also is given an option to display a spectrogram generated from the waveform of the speech utterance.

Each time the user clicks the the "Speak" button, the app will generate a new utterance. Because we randomly select from the multiple audio examples available for each IPA symbol, each generated utterance will have a slightly different pronunciation, even when resynthesizing the same word.

[6] https://streamlit.io/

6 Discussion

Over the last two centuries of studies, linguists have meticulously reconstructed the Proto-Indo-European language. Since we may never know exactly what PIE sounded like, we can only estimate its pronunciation. Confounding this issue, there are multiple and competing hypotheses in the literature debating the language's pronunciation.

The quality of synthesized speech is often evaluated using human subjects and Mean Opinion Score tests (Streijl et al., 2016). Such an evaluation approach is not feasible for a reconstructed language which lacks consensus in its phonological interpretation. Nor can we use objective tests to compare the synthesized speech to spoken examples, since no such recordings exist. For these reasons, we do not attempt empirical evaluation of the quality of the naturalness of our speech. Instead we present this flexible and interactive tool as a way to estimate hypothetical pronunciations of PIE.

6.1 Limitations

Our system is a concatenative speech synthesizer using speech sounds generated by formant synthesis. Given the limitations of these older technical approaches, our synthesized speech will not sound naturalistic. Although our generated speech may sound mechanical and emotionless, it is highly intelligible. In this work, our goal is not naturalistic speech but to provide a means to realize hypothetical speech using custom pronunciations specified by the user. Our phonetic concatenative approach yields this flexibility whereas other models, whether they be articulation synthesizers or cutting-edge deep-learning approaches, cannot.

In our process, we naïvely decontextualize phonemes from their use in their source languages and reappropriate them as individual phonetic sounds towards our goal of approximating speech in PIE. Because we are divorcing individual phonetic sounds from their pronunciation context, we are ignoring the subtle differences in the pronunciation of the same IPA symbol in different languages. For this reason, the timing or transition of some generated sequences of phones occasionally do not flow naturally together, such as an undue pause between sounds. If a pronunciation is not satisfactory, the user can readily generate another rendering.

We do not make attempts to control for other high-level aspects of the pronunciation, such as emotion or prosody. Any attempt to define rules

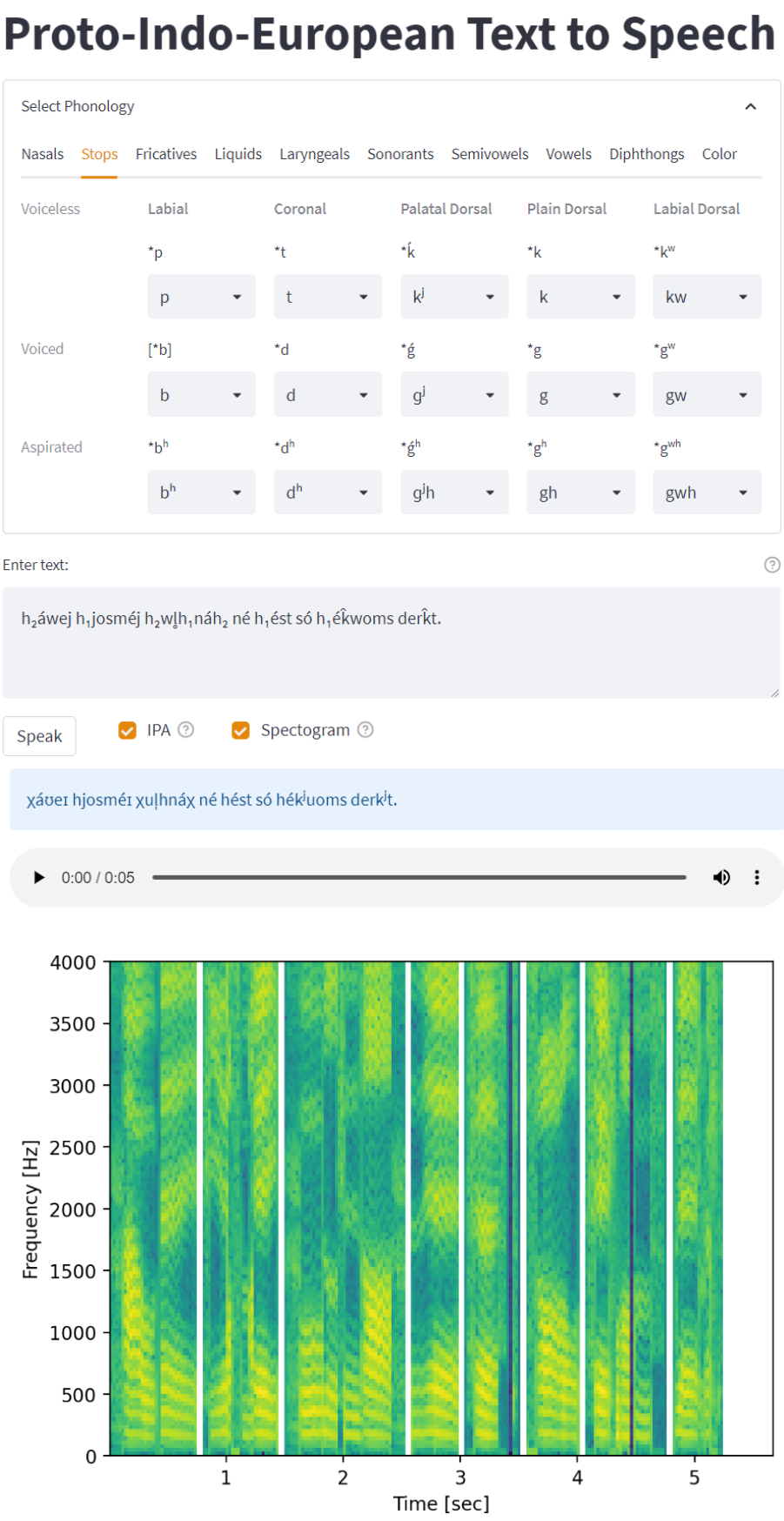


Figure 1: Screenshot of the user interface.

to control prosody in PIE would largely be based on conjuncture. We caution that attempts to make more our synthesizer sound more naturalistic would introduce even more bias, such as favoring the phonology of one language over another.

Although we sample from a large set of the world's popularly spoken languages, we are still missing several of phonetic sounds favored in some hypotheses of PIE phonology. One such sound, the voiced uvular plosive consonant represented in IPA as /ɢ/, is quite rare and occurs in only 2% of the world's languages.[7] Furthermore, articulation variants, such as /ɢʷ/, /ɢʰ/, and /ɢʷʰ/ are even more rare. We also lack examples of the labialized voiceless velar plosive /kʷ/. To compensate for these sounds we instead needed to substitute consecutive phonetic pairs from the same source utterance.

6.2 Future Work

To increase our phonetic inventory, we will consider larger words lists. This will allow us to encounter more naturally occurring sequences of two and three consecutive phones. However, increasing our inventory will also increase the disk space required to store the individual audio excerpts. Therefore, we will strategically sample from this data set to reduce excessive duplicates. Specifically, we will consider the phones that precede and follow as well as the length of a phone when deciding whether to retain or prune a phone from our dataset. In this way, we can retain a diversity of pronunciations and contexts, while eliminating those that are essentially duplicates.

As another strategy to improve our system, we will explore di-phone concatenative synthesis. A di-phone-based approach prioritizes concatenating sounds that cover the transition between individual phones. This typically consists of units sounding from the middle of a phone to the middle of the next phone. For those transitions not available in our dataset, we will substitute a single phone.

To improve our system's ability to realize all possible hypotheses of PIE phonology, we need to create examples of those few sounds we lack in our current approach. To do so, we intend to design and compile a `espeak-ng` speaker that uses the phonology we currently lack, such as the sound /ɢ/.

Although our current approach provides a few naïve options for pronunciation of PIE's accent, more work remains to improve this feature. We will consult with Indo-European linguistics to determine realistic parameters to more faithfully replicate the pitch and stress accents. We intend to add support for a rising accent, another hypothetical possibility for PIE's treatment of accent.

6.3 Contributions and Novelty

We describe a novel approach for a concatenative TTS synthesizer for the Proto-Indo-European language. We combine formant and concatenative synthesis to simulate a phonology of sounds that are not present together in any single language. When generating speech, we search among the multiple stored examples for each phone and consider the immediate phonetic context.

We present this unique and novel tool with the hopes of educating users about the ancestor language of 46% of the world's speakers. The approach presented here can be adapted by researchers to explore different hypothetical pronunciations of other reconstructed or extinct languages. For example, we envision extensions to this work to provide learners a way to aurally explore the effects of different sound laws, such as Grimm's law (Germanic), Burgmann's law (Indo-Iranian), or Winter's law (Balto-Slavic).

7 Conclusion

We propose a novel text-to-speech system to realize various hypothetical pronunciations of the Proto-Indo-European language. We synthesize speech waveforms from word lists using 100 different languages to extract a large library of individual phonetic sounds. Using this inventory, we build a concatenative speech synthesizer that combines phones in an attempt to recreate spoken speech in PIE. Since the precise pronunciation of many phonemes in PIE is uncertain, our system provides a user interface that permits users to select a specific IPA realization for each PIE phoneme and diphthong. Where available in the data, we prioritize extracting consecutive phones from the same original source utterance in order to provide continuity of the waveform and the pronunciation. We randomly select from the matching phones to provide a slightly different pronunciation each time a text is synthesized. We add small randomized pauses between syllables, words, and sentences to better emulate naturalistic speaking rates. We provide an interactive demonstration of our tool at `https://soundbendor.org/projects/PIE`.

[7] `https://phoible.org/`

References

Douglas Q Adams. 1997. *Encyclopedia of Indo-European Culture*. Taylor & Francis.

Sercan Ö Arık, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al. 2017. Deep voice: Real-time neural text-to-speech. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 195–204.

Charles M Barrack. 2002. The glottalic theory revisited: a negative appraisal. *Indogermanische Forschungen*, 107(1):76–95.

Robert SP Beekes. 2011. *Comparative Indo-European linguistics: an introduction*. John Benjamins Publishing.

Laurent Besacier, Etienne Barnard, Alexey Karpov, and Tanja Schultz. 2014. Automatic speech recognition for under-resourced languages: A survey. *Speech communication*, 56:85–100.

Andrew Miles Byrd. 2018. 121. the phonology of proto-indo-european. In *Volume 3 Handbook of Comparative and Historical Indo-European Linguistics*, pages 2056–2079. De Gruyter Mouton.

Ailbhe Ní Chasaide, Neasa Ní Chiaráin, Christoph Wendler, Harald Berthelsen, Andy Murphy, and Christer Gobl. 2017. The abair initiative: Bringing spoken irish into the digital space. In *Proc. of Interspeech*, pages 2113–2117.

Vlado Delić, Zoran Perić, Milan Sečujski, Nikša Jakovljević, Jelena Nikolić, Dragiša Mišković, Nikola Simić, Siniša Suzić, and Tijana Delić. 2019. Speech technology progress based on new machine learning paradigm. *Computational intelligence and neuroscience*.

Marija Gimbutas. 1956. *The prehistory of eastern Europe*. 20. Harvard University.

Henry M Hoenigswald. 1963. On the history of the comparative method. *Anthropological linguistics*, pages 1–11.

Jay Jasanoff. 2017. The impact of hittite and tocharian: Rethinking indo-european in the 20th century and beyond. *Handbook of Comparative and Historical Indo-European Linguistics*, pages 220–238.

Oliver Jokisch and Matthias Eichner. 2000. Synthesizing and evaluating an artificial language: Klingon. In *Proc. 6th International Conference on Spoken Language Processing (ICSLP 2000)*, volume 1, pages 729–732.

Mate Kapović. 2017. Proto-indo-european phonology. *The Indo-European Languages*, pages 13–60.

Allan R Keiler. 2015. A phonological study of the indo-european laryngeals. In *A Phonological Study of the Indo-European Laryngeals*. De Gruyter Mouton.

Rubeena A Khan and JS Chitode. 2016. Concatenative speech synthesis: A review. *International Journal of Computer Applications*, 136(3):1–6.

Sneha Lukose and Savitha S Upadhya. 2017. Text to speech synthesizer-formant synthesis. In *2017 International Conference on Nascent Technologies in Engineering (ICNTE)*, pages 1–4. IEEE.

Mario Malcangi and Philip Grew. 2010. Toward language-independent text-to-speech synthesis. *WSEAS Transactions on Information Science and Applications*, 7(3):411–421.

Michael Meier-Brügger. 2013. *Indo-European Linguistics*. de Gruyter.

Yishuang Ning, Sheng He, Zhiyong Wu, Chunxiao Xing, and Liang-Jie Zhang. 2019. A review of deep learning based speech synthesis. *Applied Sciences*, 9(19):4050.

Perez Ogayo, Graham Neubig, and Alan W Black. 2022. Building african voices. *arXiv preprint arXiv:2207.00688*.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Charles E. Prescott. 2018. Pharyngealization and the three dorsal stop series of proto-indo-european. In *The Meaning of Language*, page 220. Cambridge Scholars Publishing.

Robert C Streijl, Stefan Winkler, and David S Hands. 2016. Mean opinion score (mos) revisited: methods and applications, limitations and alternatives. *Multimedia Systems*, 22(2):213–227.

Morris Swadesh. 1952. Lexico-statistic dating of prehistoric ethnic contacts: with special reference to north american indians and eskimos. *Proceedings of the American philosophical society*, 96(4):452–463.

P Swiggers. 1989. Towards a characterization of the proto-indo-european sound system. In Theo Vennemann, editor, *The New Sound of Indo-European: Essays in Phonological Reconstruction*, pages 177–208. De Gruyter Mouton.

Daniel R Van Niekerk and Etienne Barnard. 2009. Phonetic alignment for speech synthesis in under-resourced languages. In *Proc. of Interspeech*, pages 880–883.

Junichi Yamagishi, Takashi Nose, Heiga Zen, Zhen-Hua Ling, Tomoki Toda, Keiichi Tokuda, Simon King, and Steve Renals. 2009. Robust speaker-adaptive hmm-based text-to-speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1208–1230.

A low latency technique for speaker detection from a large negative list

Yu Zhou
Vail Systems, Inc.
yzhou@vailsys.com

Vijay K. Gurbani
Vail Systems, Inc.
vgurbani@vailsys.com

B. Chandra Mouli
Vail Systems, Inc.
chandra@vailsys.com

Abstract

Negative list (NL) detection, commonly referred as open-set multi-target speaker detection, attempts to match a test utterance with any one of a set of known utterances enrolled in the negative list. A number of normalization techniques have been developed for similarity score calibration in order to increase the detection accuracy. While these normalization methods apply to both single-target verification and multi-target detection, in this work we propose NL-Norm, a novel normalization technique that is designed specifically for multi-target detection by considering scores between all enrolled NL utterances and the normalization cohort as a single distribution. Furthermore, we propose using Locality Sensitive Hashing (LSH) to efficiently find a small subset of utterances from enrolled NL utterances and the normalization cohort that are most similar to the test utterance, so that the number of similarity score computations can be significantly reduced. The combination of these novel techniques is evaluated on the MCE 2018 datasets. Applying LSH and NL-Norm, our approach demonstrated significant improvements in both speed and accuracy over using PLDA only in the backend, resulting in 88% reduction in detection time while decreasing the equal-error rate (EER) from 6.49% to 5.48% for MCE2018 dataset.

1 Introduction

Much progress has been made in speaker recognition, as demonstrated by continuously improving results in US National Institute of Standards and Technology (NIST) Speaker Recognition Evaluation series (Greenberg et al., 2020; Sadjadi et al., 2020). Meanwhile, in real world applications such as call center services, Negative List (NL) detection is often a crucial component of fraud detection based on voice biometrics. In NL detection, a call is flagged for investigation if an utterance is determined to be spoken by one of the known fraudulent speakers that are enrolled in the NL, without needing to identify which specific NL speaker the caller is matched with.

NL detection is often referred as open-set multi-target speaker detection. While most speaker recognition studies focus on the single-target problem, in recent years there have been efforts to develop methods used for multi-target speaker recognition (Singer and Reynolds, 2004; Zigel and Wasserblat, 2006; Malegaonkar and Ariyaeeinia, 2011; Gunson et al., 2015) For example, MCE 2018 (Shon et al., 2019) is a challenge designed specifically to promote methods for multi-target cohort detection and multi-target identification: the former, also known as Top-S detction, is aimed to only detect whether the input speech is spoken by a member of the NL cohort; the latter, referred as Top-1 detection, not only detects membership in the NL cohort but further identifies the specific speaker within the NL. In this paper we focus on the Top-S detection only, and use the call center industry term Negative List detection to distinguish it from Top-1 detection or open-set speaker identification.

The techniques used by NL detection are mostly common with that of single-target speaker recognition and are depicted in Figure 1. During the training phase (Figure 1(a)), a front-end Gaussian Mixture Model with Universal Background Model (GMM/UBM) (Reynolds et al., 2000; Dehak et al., 2011)) or a Deep Neural Network (DNN) model

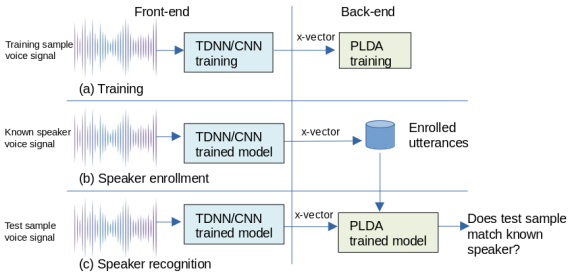


Figure 1: Steps in speaker identification

(Snyder et al., 2018) is trained to extract the speaker embedding i-vector or x-vector, respectively. (For brevity, Figure 1 only shows x-vector.) Subsequently, a back-end model such as Probabilistic Linear Discriminant Analysis (PLDA) is trained to produce a matching score between a pair of input i-vectors or x-vectors representing the log-likelihood ratio of the two vectors belonging to the same speaker.

Next, during the speaker enrollment phase (Figure 1(b)), i-vectors or x-vectors are extracted from a known speaker's utterances and stored in the database as voice biometric signature of that speaker. Finally, for speaker recognition (Figure 1(c)), the i-vector or x-vector extracted from an out-of-sample test utterance is paired with the enrolled i-vector or x-vector of a known speaker for the PLDA model to compute the matching score in order to determine whether the test utterance matches the enrolled speaker (Prince and Elder, 2007). Score normalization (Fortuna et al., 2004, 2005; Zigel and Wasserblat, 2006; Matějka et al., 2017) is typically performed on the PLDA output before it is compared with a pre-determined threshold for the match/no-match decision.

Since single-target and multi-target speaker recognitions share the same techniques described above, a multi-target speaker recognition problem can be effectively treated as multiple single-target recognitions. However, for call center services, there are often thousands of fraudulent speakers in the NL, which poses some unique challenges: with the increase in the NL size, the detection error as measured by equal error rate (EER) becomes higher (Shon et al., 2019), and the computing cost in the form of detection latency for each test grows as well. These challenges impede an effective real-time NL detection implementation. In this work, we propose enhancements to back-end processing for multi-target applications with the twin objectives of lowering EER and achieving a faster time for NL detection in the face of an increasing number of speakers in the NL list.

Our contributions are:

- **A normalization technique (NL-Norm) to calibrate consistent scores for NL detection.** We describe a normalization technique that considers similarity scores between the normalization cohort and all enrolled NL speakers as a single distribution; this helps to better calibrate test scores and select a consistent threshold for NL detection;
- **Reduction of the computation cost at inference time.** Locality Senstive Hashing (LSH (Indyk and Motwani, 1998)) is used to find a subset of NL speakers and a subset of the normalization cohort that are most similar to the test utterance, so that the similarity scores are evaluated between the test utterance and these two subsets only, which reduces the computation cost at inference time. The advantage of LSH is further amplified when different adaptive lengths for Z-Norm and T-Norm are allowed within AS-Norm.

2 Preliminaries

In existing literature, the PLDA model is often combined with score normalization to create an effective backend processing of a speaker recognition system (Fortuna et al., 2004, 2005; Zigel and Wasserblat, 2006; Matějka et al., 2017). Once the PLDA score, $s(e,t)$, between a test utterance t and an NL-enrolled utterance e is generated, various normalization techniques can be applied for score calibration in order to derive a consistent matching threshold. For NL detection, t is determined as an NL match if the highest normalized score between t and all NL utterances is above the threshold; this results in a Top-S set from which the EER metric is drawn to measure the accuracy of the model (Shon et al., 2019; Singer and Reynolds, 2004).

2.1 Score Normalization

All score normalizations require a normalization cohort consisting utterances from speakers that are neither in NL nor part of the test cohorts. We denote NL-enrolled utterance set and the normalization cohort as $\mathcal{E}$ and $\mathcal{C}$, respectively:

$$\begin{aligned} \mathcal{E} &= \{e_i \mid 1 \le i \le E\} \\ \mathcal{C} &= \{c_i \mid 1 \le i \le N\} \end{aligned} \tag{1}$$

where E and N are the size of NL and the normalization cohort, respectively.

Z-Norm (Li and Porter, 1988) utilizes the scores between an NL speaker utterance e and every utterance c_i in the normalization cohort:

$$S_e = \{s(e,c_i) \mid 1 \le i \le N\} \tag{2}$$

resulting in the normalized score:

$$s(e,t)_{z-norm} = \frac{s(e,t) - \mu(S_e)}{\sigma(S_e)} \tag{3}$$

where $\mu(S_e)$ and $\sigma(S_e)$ are the mean and standard deviation of S_e, respectively. For NL detection, Eq.(2) needs to be evaluated for every enrolled speaker $e \in \mathcal{E}$. However, these evaluations can be carried out during NL enrollment instead of at inference time. At each NL detection, the most computationally expensive task is to calculate E similarity scores $\{s(e_i, t) \mid 1 \leq i \leq E\}$, which is required regardless of score normalization.

T-Norm (Auckenthaler et al., 2000) uses the scores between t and every utterance c_i in the normalization cohort:

$$S_t = \{s(t, c_i) \mid 1 \leq i \leq N\}$$
$$s(e,t)_{t-norm} = \frac{s(e,t) - \mu(S_t)}{\sigma(S_t)} \quad (4)$$

In contrast to Z-Norm, Eq.(4) is evaluated at inference time for a given t, therefor there are $(N + E)$ similarity scores to be computed at each NL detection using T-Norm.

AS-Norm (Karam et al., 2011; Cumani et al., 2011) is often found to have the best performance, especially for multi-target recognitions. It is defined as the average of adaptive Z-Norm and T-Norm, namely, for an adaptive length K:

$$s(e,t)_{as-norm} = \frac{1}{2}\left(\frac{s(e,t) - \mu(S_e^{(K)})}{\sigma(S_e^{(K)})} + \frac{s(e,t) - \mu(S_t^{(K)})}{\sigma(S_t^{(K)})}\right) \quad (5)$$

where $S_e^{(K)}$ and $S_t^{(K)}$ denote the subsets consisting of the highest K scores in S_e and S_t, respectively. For NL detection, S_e (thus $S_e^{(K)}$) can be evaluated as soon as the NL is constructed, however S_t (thus $S_t^{(K)}$) can only be calculated when the test utterance t is present. Therefore at inference time for each t the number of similarity scores to be generated is $(N + E)$.

2.2 MCE 2018 dataset description

The work reported in this paper is conducted on the MCE 2018 dataset, a public dataset curated from recordings of customer-agent conversations to an operational call center (Shon et al., 2019). The dataset, consisting of negative list speakers and background speakers (i.e. not on the negative list), is summarized in Table 1.

The MCE 2018 dataset is provided in the form of i-vectors corresponding to each of the negative list and background speaker utterances. Using this dataset has several advantages: NL detection (Top-S detection) is one of the tasks in MCE challenge;

Table 1: MCE 2018 dataset description

Set	Subset	No. of speakers	Total utterances
Train	Negative list	3,631	10,893
	Background	500	30,952
Dev.	Negative list	3,631	3,631
	Background	5,000	5,000
Test	Negative list	3,631	3,631
	Background	12,386	12,386

the dataset consists of 600-dimension i-vectors extracted from call center conversations, the domain of interest in our study; and the large number of enrolled NL speakers E = 3631 is within range of real-world NL sizes. In addition, the techniques investigated in this work are part of the speaker recognition backend process, therefore using a set of i-vectors that has been validated by previous studies (Khoury et al., 2019; Font, 2019; Wilkinghoff, 2020) eliminates the need for vector extraction, and prevents introducing unnecessary variabilities for the purpose of this work.

2.3 Using LSH for low-latency search at inference time

PLDA model is the preferred method for similarity score generation due to its accuracy as measured in EER (Prince and Elder, 2007; Matejka et al., 2011). However, PLDA is computationally expensive for NL detection as it requires computing a large number of pairwise scores to determine the membership of a test utterance in the NL set. While Linear Discriminant Analysis (LDA) (Matejka et al., 2011) can be applied prior to PLDA to reduce dimensions and speed up computation of the score, the limiting factor will be the size of the NL. In commercial call-center applications it is not uncommon for the NL to contain thousands of entries, making the PLDA approach unfeasible for real-time detection.

To speed up the search, we propose using LSH, a family of functions used to solve the nearest neighbor problem by finding approximate — instead of exact — matches (Indyk and Motwani, 1998). LSH hashes the data and a query point in a way that maximizes the probability of a collision for points that are close to each other than for those which are farther apart. This approximation allows efficient solutions to exist when the dimensionality, m, is large. Further, Indyk and Motwani (Indyk and Motwani, 1998) show that LSH requires $O(mn^{1+1/c})$ processing time and $O(mn^{1/c})$ query time, where c is a constant that constrains the radius around which points should match. A crucial parameter in LSH is the choice of a distance function; exist-

ing literature (Charikar, 2002; Schmidt et al., 2014) demonstrates that the cosine similarity measure can be approximated well with locality sensitive hash functions.

LSH minimizes run time at the expense of accuracy, however, as our results in Section 6 demonstrate, the approximate matches retrieved by LSH have a high probability of being correct as reflected in the lowered EER.

3 Related Work

Open-set speaker recognition techniques using PLDA are often enhanced with score normalization. Li and Porter (Li and Porter, 1988) propose Z-Norm, which normalizes the PLDA score between the test and target utterances to the distribution of scores between the target and a normalization cohort. Auckenthaler et al. (Auckenthaler et al., 2000) present T-Norm, which applies the score distribution between the test utterance and the normalization cohort. S-Norm (Kenny, 2010) is defined as the average of Z-Norm and T-Norm, while Karam et al. (Karam et al., 2011) and Cumani et al. (Cumani et al., 2011) suggest AS-Norm, which constrains the S-Norm computations to subsets that contain the highest normalization scores from the T-Norm and the Z-Norm in order to produce the most relevant PLDA score distributions (Eq.(5)).

Our contribution, NL-Norm, is fashioned after AS-Norm with one distinction unique to the Negative List detection. Instead of normalizing to the score distribution of a single target utterance as Z-Norm, NL-Norm constructs the normalization distribution using the collection of PLDA scores between the normalization cohort and all enrolled utterances in the NL. In addition, by allowing different adaptive lengths for T-Norm and (modified) Z-Norm terms in AS-Norm and NL-Norm, we can take full advantage of LSH for optimal speed and accuracy in NL detection. The detailed description is provided in Section 4.

There exists a large body of literature on the use of LSH for audio data, especially in discovering similar songs and de-duplicating remixes. However, our literature review restricts itself to those works that use LSH in the context of NL detection. The most closely related work on using LSH for NL detection is Schmidt et al. (Schmidt et al., 2014), which uses LSH to quickly approximate the cosine distance in their retrieval process. They eschew the use of PLDA because "this method performs a more complicated hypothesis for i-vector matching, which impedes its use with LSH." (Schmidt et al., 2014). While the computationally expensive nature of PLDA precludes its use as a distance function in LSH, our work demonstrates that PLDA can nonetheless be used effectively by allowing LSH to constrain the number of PLDA operations required to determine a match from the NL. Ma et al. (Ma et al., 2015) create clusters of low dimensional i-vectors and restrict PLDA score computation to the top-n closest clusters. They compare their approach to LSH, but unlike our work, they do not use LSH to restrict the PLDA score computations to a small set. Other surveyed works use LSH, however, unlike our work, they do not consider in in the context of PLDA computations: Jeon et al. (Jeon and Cheng, 2012) use kernelized LSH for speaker identification, while Leary et al. (Leary and Andrews, 2014) substitute the Hamming distance for cosine distance in LSH; and finally, Shon et al. (Shon et al., 2018) use random projects generated from a speaker variability space to derive the characteristic matrix in LSH.

4 Negative List Specific Techniques

In this section we propose novel techniques devised for NL detection, including: using NL-Norm, a NL-specific score normalization, to improve detection accuracy; taking advantage of different adaptive lengths for S_e and S_t in AS-Norm (Eq.(5)) and NL-Norm; and applying LSH in conjunction with NL-Norm to reduce detection latency at inference time.

4.1 NL-Norm

Z-Norm in Eq.(3) is designed to compensate similarity score variations against a single target speaker e, so that the scores between e and all test utterances can be normalized to the same distribution in order to apply a consistent threshold for speaker verification. For NL detection, since all enrolled speaker utterances are present at testing time, and a single threshold for normalized scores is needed for all speakers in NL, it is reasonable to introduce a normalization over the entire NL cohort which we refer as NL-Norm. Formally, the scores used for NL-Norm consists of pairwise scores between every normalization cohort member c_i and every NL member e_j:

$$S_{NL} = \{S_{e_j} \mid 1 \leq j \leq E\} \tag{6}$$

where S_{e_j} is a rewrite of Eq.(2):

$$S_{e_j} = \{s(e_j, c_i) \mid 1 \leq i \leq N\} \tag{7}$$

Practically, we define NL-Norm with an adaptive length K that is analogous to AS-Norm in Eq.(5):

$$s(e,t)_{nl-norm} = \frac{1}{2}\left(\frac{s(e,t)-\mu(S_{NL}^{(K)})}{\sigma(S_{NL}^{(K)})} + \frac{s(e,t)-\mu(S_t^{(K)})}{\sigma(S_t^{(K)})}\right) \quad (8)$$

where

$$S_{NL}^{(K)} = \{S_{e_j}^{(K)} \mid 1 \le j \le E\} \quad (9)$$

with $S_{e_j}^{(K)}$ denoting the subset consisting of the highest K socres in S_{e_j}.

4.2 Different Adaptive Lengths

Even though Eq.(5) is commonly used for AS-Norm, there is no theoretical constraint that the same adaptive length K must be used for Z-Norm and T-Norm terms. By allowing different adaptive lengths K_e and K_t for Z-Norm and T-Norm, respectively, Eq.(5) can be modified as:

$$s'(e,t)_{as-norm} = \frac{1}{2}\left(\frac{s(e,t)-\mu(S_e^{(K_e)})}{\sigma(S_e^{(K_e)})} + \frac{s(e,t)-\mu(S_t^{(K_t)})}{\sigma(S_t^{(K_t)})}\right) \quad (10)$$

Similarly, Eq.(8) for NL-Norm can be modified with different adaptive lengths K_t and K_e:

$$s'(e,t)_{nl-norm} = \frac{1}{2}\left(\frac{s(e,t)-\mu(S_{NL}^{(K_e)})}{\sigma(S_{NL}^{(K_e)})} + \frac{s(e,t)-\mu(S_t^{(K_t)})}{\sigma(S_t^{(K_t)})}\right) \quad (11)$$

While employing different K_e and K_t in AS-Norm is not a novel approach (it was previously used in (Wilkinghoff, 2020)), here we explicitly explore its advantages when applied to NL detection: decoupling K_e and K_t not only enables further optimization of detection accuracy, it also allows the selection of a small K_t value without sacrificing the benefit of Z-Norm that may require a larger K_e. The combination of LSH with a small K_t can significantly improve the speed of NL detection by reducing the number score computations between the test utterance t and the normalization cohort, which is described further in the next section.

4.3 Reducing inference latency through LSH

In NL detection, for a given test utterance t, its similarity scores with all members of the NL are computed and ranked, with or without score normalization. The top ranked score is then compared with the pre-determined threshold to reach a decision. The search time here will be dominated by $O(E)$, where $E = |\mathcal{E}|$.

As discussed in Section 2.3, LSH is used to speed up the search process. For the work described here, this means the following: First, the LSH pipeline is "trained" on the i-vectors or x-vectors associated with the $\mathcal{E}$ and $\mathcal{C}$. (Recall from Section 2 that $\mathcal{C}$ is the normalization cohort.) Here, "training" implies deriving shorter characteristic vector for each of the 600-dimension vectors in $\mathcal{E}$ and $\mathcal{C}$ using a set of random hyperplane-based hash functions (Charikar, 2002). Given a collection of vectors in $\mathbb{R}^m$, we choose a random vector $\vec{r}$ from the $m-$dimension Gaussian distribution and define a hash function $h_{\vec{r}}$ as follows:

$$h_{\vec{r}}(\vec{u}) = \begin{cases} 1 & \vec{r}\cdot\vec{u} \ge 0 \\ 0 & \vec{r}\cdot\vec{u} < 0 \end{cases} \quad (12)$$

Then, for any vectors $\vec{u}$ and $\vec{v}$,

$$\mathbf{Pr}[h_{\vec{r}}(\vec{u}) = h_{\vec{r}}(\vec{v})] = 1 - \frac{\theta(\vec{u},\vec{v})}{\pi} \quad (13)$$

where $\theta(\vec{u},\vec{v})$ is the angle between vectors u and v. Further, for n vectors, the hash functions can be chosen by picking $O(log^2 n)$ random bits, thereby restricting the random hyperplanes to be in a family of size $2^{O(log^2 n)}$ (Charikar, 2002). For a given test i-vector t, we use LSH twice: once to discover K_e nearest neighbors to t from $\mathcal{E}$, and the second time to discover K_t nearest neighbors to t from $\mathcal{C}$ (the K_t mentioned in Section 4.2). Therefore, in Eq.(10) and Eq.(11), $S_t^{(K_t)}$ is constructed by identifying K_t members in $\mathcal{C}$ using LSH followed by the generation of K_t scores, instead of generating all N scores followed by the identification of top K_t scores. (Recall that $N = |\mathcal{C}|$.) Using such an approach, for each t, the number of PLDA score evaluations is reduced from $O(E+N)$ to $O(K_e + K_t)$. Because LSH search time is negligible comparing with PLDA score evaluation, this approach can significantly reduce the computational cost and latency of NL detection.

5 Experimental Setup

The training set of MCE dataset, consisting of 3,631 NL speakers and 5,000 background (non-NL) speakers, is used for PLDA model training. There are three utterances from each NL speaker, the mean of the three i-vectors is enrolled in NL as the utterance of the corresponding fraudster. Furthermore, similar to Khoury et al. (Khoury et al.,

2019), a normalization cohort of 4,000 augmented i-vectors is generated by applying at random a weighted sum between non-NL speaker i-vectors and NL speaker i-vectors, limiting the maximum weight for NL speakers to 20%. The Development set, consisting of one utterance from each of 3,631 NL speakers and 5,000 non-NL speakers, is used to verify the baseline approach of PLDA with AS-Norm, including the tuning of adaptive length K in AS-Norm for baseline EER computation on the Test set.

A stratified 50/50 random split of MCE Test set, consisting of one utterance from each of 3,631 NL speakers and 12,386 non-NL speakers, produces equal-sized Validation set and Evaluation set. The Validation set is used for tuning of hyper-parameters including LSH depth L, adaptive length K in NL-Norm Eq.(8), and K_t and K_e in Eqs.(10) and (11). Evaluation set is reserved for holdout testing only. Unless specified otherwise, all results presented in this paper are obtained using the Evaluation set. The motivation for generating the Validation and Evaluation sets from MCE Test set is that the 50/50 stratified split preserves the ratio of non-NL to NL speakers of the Test set, which is significantly higher than that of the Development set. It is worth noting that in real-world call center applications this ratio is much higher (Khoury et al., 2019). In addition, it is desirable that Validation and Evaluation sets have similar distributions and behaviors, whereas MCE Development set exhibits much lower EER than the Test set (Shon et al., 2019).

In contrast to MCE 2018 Challenge participating studies where the goal is to achieve the lowest EER, the aim of this work is to examine the effectiveness of the novel NL detection techniques outlined in Section 4. Therefore we adopt a minimal baseline approach: PLDA followed by AS-Norm, in order to remove potential variations introduced by nonessential steps. For example, LDA is eliminated even though it is often used prior to PLDA for dimension reduction. With this baseline, we observed NL detection EER of 1.25% and 5.66% for MCE Development and the entire Test set, respectively, which are comparable to published results in MCE 2018 Challenge (Shon et al., 2019).

PLDA implementation in the Kaldi toolkit (Povey et al., 2011) is used to generate similarity scores $s(e,t)$, $s(e,c_i)$ and $s(t,c_i)$. After score normalization, for each test utterance t the highest score among its normalized scores with NL utterances is used for the overall EER calculation (Singer and Reynolds, 2004). In our work, LSH is applied prior to PLDA to reduce the number of similarity score computations. We use the NearPy Python framework[1] as the LSH implementation with random hyperplane-based hash functions.

All computation times (i.e., detection latency) indicated in this paper were measured on an Intel Core i7-7700HQ 2.8GHz CPU with 16GB RAM.

6 Results and Discussion

6.1 NL-Norm

Applying Eq.(8) of NL-Norm, the resulting EER of NL detection is 5.57%, reduced from 5.69% for AS-Norm. To verify the stability and consistency of this result, a cross validation is performed by repeating 10 times the stratified random 50/50 split of MCE Test set into validation and evaluation sets, the outcome is shown in Table 2, where an average of 0.11% reduction in EER absolute value is observed.

Table 2: NL-Norm vs AS-Norm EER

		AS-Norm	**NL-Norm**
EER (Evaluation Set)		5.69%	5.57%
EER (Cross Validation)	mean	5.64%	5.53%
	std	0.26%	0.28%

It is worth noting that because NL-Norm considers the scores between the normalization cohort and the entire NL cohort as a single distribution, it can diminish the distinctions among individual NL speakers. Therefore we postulate that NL-Norm may not improve multi-target speaker identification (i.e. Top-1 identification). This is beyond the current NL detection study and remains to be examined in the future.

6.2 Adaptive Length K_e and K_t

Comparing AS-Norm of Eq.(10) with Eq.(5), with the additional tuning parameter, a lower EER can be reached when separate adaptive lengths K_e and K_t are employed. Table 3 shows the tuning progress on the Validation Set, where the first row represents AS-Norm that requires $K_e = K_t$, and the second row represents the best result found with $K_e = 1600$ and $K_t = 600$. In addition to the gain in accuracy, more importantly, this approach offers the flexibility of selecting a low K_t value that yields a close-to-optimal EER, as demonstrated by the last row of Table 3, where $K_t = 200$. A low K_t,

[1] https://github.com/pixelogik/NearPy

combined with LSH, enables a significant reduction in the number of score computations between the test utterance t and the normalization cohort, which in turn increases the NL detection speed. The detail is presented in the next section.

Table 3: AS-Norm with Different Adaptive Lengths

K_e	K_t	EER (%)
300	300	5.69
1600	600	5.61
3800	200	5.62

6.3 The utility of LSH

Consider a test vector, t, which must be compared against a NL of size $E = 3,631$. As Table 4 shows, the fastest distance algorithm for comparison is cosine distance, which takes 4ms to compare t against all of the NL entries. However, its speed is achieved at the cost of accuracy: the cosine distance yields an EER of 7.40%. PLDA improves on the EER but at the expense of an increased latency. LSH followed by PLDA allows us to not only derive an EER similar to that of PLDA only, but it also does so at a fraction of time: 28ms compared to 864ms for PLDA. For the test vector t, LSH search is first conducted to find L members of NL that are most similar to t, then PLDA scores between t and these L utterances are computed, with the highest score selected for EER calculation.

Table 4: Accuracy and Latency

Algorithm	**EER (%)**	**Time (ms)**
Cosine distance	7.40	4
PLDA	6.49	864
LSH + PLDA (L=30)	6.46	28

When the LSH depth L increases, it is expected the resulting EER will approach the EER exhibited by PLDA if a greedy score calculation strategy is used across the entire NL. Such an expectation is demonstrated empirically in Figure 2, where the red line represents the EER obtained from PLDA (without score normalization). For the NL dataset with size $E = 3,631$, an LSH depth $L = 30$ is already sufficient to match the EER of PLDA only.

It should be noted that even though EER $= 6.46\%$ at $L = 30$, which is slightly better than the EER of 6.49% without LSH, it is not an indication that LSH can help improve NL detection accuracy. The reason for the occasionally lower EER when performing LSH first is that for a test utterance not spoken by any NL speaker, LSH may fail to find the NL member that would have produced the highest PLDA score, thus eliminating a false-match instance from the EER evaluation.

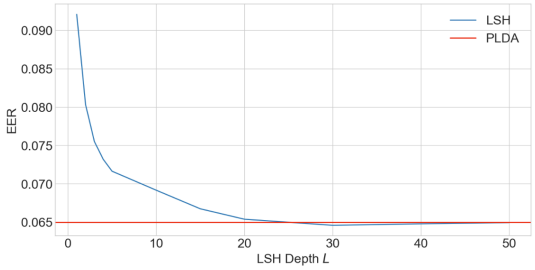


Figure 2: EER vs.LSH depth L without score normalization

As seen in Fig. 2, this effect quickly disappears with increasing LSH depth L. The results in Fig. 2 are obtained without score normalization. Score normalization improves EER, we thus examine the effect of combining LSH, PLDA and score normalization next.

6.4 Putting it together: LSH and NL-Norm

Algorithm 1 presents our use of LSH and NL-Norm. Lines 1-12 establish the normalization over the entire NL cohort as discussed in Section 4.1; this step is done only once, during initialization. Each enrolled speaker's utterances are scored against all of the normalization cohorts and the top K_e matches for that speaker and the normalization cohort are saved (line 10). At the end of the loop on line 11, $S_{NL}^{(K_e)}$ is populated and will be used to compute the NL-Norm later. As part of populating $S_{NL}^{(K_e)}$, it is possible that the eventual top-ranked NL speaker which induces the highest normalized PLDA score is not among the top K_e NL members identified by the LSH search, resulting in a decrease in the NL detection accuracy. This deficiency can be mitigated by expanding LSH search depth K_e. Line 12 invokes the NL-Norm computation function that returns true if a match is found.

For a given target vector t, we first find the top-L nearest neighbours of t from the NL $\mathcal{E}$ (line 14). To apply normalization, PLDA scores between t and utterances in the normalization cohort $\mathcal{C}$ are also needed; LSH can be utilitized once more to find K_t utterances in $\mathcal{C}$ that are most similar to t (line 15). By the end of the loop on line 19, both the sets $S_t^{(K_t)}$ and $S_{NL}^{(K_e)}$ are available, the latter populated during initialization as described in the preceding paragraph. Finally, lines 21-25 computes the normalized PLDA score between t and each NL list member, saving the results in a list (Line 25) that is checked against the threshold to determine an NL match. With Algorithm 1 the total

number of PLDA evaluations are reduced from $O(E+N)$ without LSH to $O(L+K_t)$ with LSH, where $L \ll E$ and $K_t \ll N$.

Algorithm 1: LSH and NL-Norm

Data: t: Test vector to be matched in $\mathcal{E}$
Data: T: Threshold for NL detection
Data: L, K_t, K_e: LSH depth and adaptive lengths, selected using Validation set (Sec. 5)
Data: $\mathcal{C}$: The normalization cohort
Data: $\mathcal{E}$: The negative list

$initialized \leftarrow False$;
if *(!initialized)* **then**
  $initialized \leftarrow True$;
  $S_{NL}^{(K_e)} \leftarrow \emptyset$;
  for $(e \in \mathcal{E})$ **do**
    $S_e \leftarrow \emptyset$;
    for $(c \in \mathcal{C})$ **do**
      $S_e \leftarrow S_e \cup score(e,c)$;
    end
    $S_{NL}^{(K_e)} \leftarrow S_{NL}^{(K_e)} \cup top\_n(S_e, K_e)$ // Eq.9
  end
$return\ NL\_norm(\mathcal{E}, \mathcal{C}, S_{NL}^{(K_e)}, t, L, K_t, T)$
function *boolean NL_norm($\mathcal{E}$: list, $\mathcal{C}$: list, $S_{NL}^{(K_e)}$: list, t: vector, L: int, K_t: int, T: float)*:
  $t\_\mathcal{E}\_set \leftarrow LSH\_lookup(t, \mathcal{E}, L)$;
  $t\_\mathcal{C}\_set \leftarrow LSH\_lookup(t, C, K_t)$;
  $S_t^{(K_t)} \leftarrow \emptyset$;
  for $(q \in t\_\mathcal{C}\_set)$ **do**
    $S_t^{(K_t)} \leftarrow S_t^{(K_t)} \cup score(t,q)$ // Eq.4
  end
  $result\_set \leftarrow \emptyset$;
  for $(e \in t\_\mathcal{E}\_set)$ **do**
    $s \leftarrow score(e,t)$;
    $s_{nl\_norm} \leftarrow \frac{1}{2}\left(\frac{s-\mu(S_{NL}^{(K_e)})}{\sigma(S_{NL}^{(K_e)})} + \frac{s-\mu(S_t^{(K_t)})}{\sigma(S_t^{(K_t)})}\right)$ // Eq.11
    $result\_set \leftarrow results\_set \cup \{s_{nl\_norm}\}$;
  end
  $return\ max(result\_set) > T ?\ True\ \ False$;

Table 5 lists the results obtained using various approaches, including the NL detection time per test utterance along with parameters L, K_e and K_t which, as described in Section 5, are selected via a grid search for the lowest EER on the Validation set, then applied to the Evaluation set . As demonstrated in Section 4.2, if multiple $\{K_e, K_t\}$ pairs yield near-lowest EERs for the Validation set, then the one with a low K_t is selected to take full advantage of LSH.

The first two rows in Table 5 are results of the baseline model, with and without score normalization. By applying LSH without score normalization, the NL detection time per test utterance is reduced from 864ms to 28ms, with little change in EER. When adopting Eq.(11) for score normalization, EER is lowered to 5.48% by taking advantage of NL-Norm and allowing different adaptive lengths K_t and K_e, at the same time the NL detection time is shortened significantly from 1975ms to 102ms, a beneficiary of LSH search. As noted previously, these results are obtained using the 600-dimension i-vector as input to PLDA model directly. A dimension reduction step such as LDA can be inserted before PLDA to reduce the inference time further for all approaches listed in Table 5, with or without LSH search. Nonetheless the use of LSH already make real-time NL detection feasible for applications such as call center services.

Table 5: EER and NL Detection Time (per utterance)

Method	EER (%)	L	K_e	K_t	Time (ms)
PLDA, no score norm	6.49	-	-	-	864
AS-Norm (Eq.(5))	5.69	-	300	300	1975
NL-Norm (Eq.(8))	5.57	-	500	500	1981
AS-Norm (Eq.(10))	5.52	-	3800	200	2051
NL-Norm (Eq.(11))	5.57	-	400	500	1976
LSH + PLDA, no score norm	6.46	30	-	-	28
LSH + AS-Norm (Eq.(10))	5.66	50	2000	250	114
LSH + NL-Norm (Eq.(11))	5.48	50	3700	200	102

7 Conclusion

Negative list detection is an important application for fraud detection in various industries such as call center services. This work explored novel techniques specifically devised for NL detection, with the aim of improving both accuracy and speed. NL-Norm considers similarity scores between the normalization cohort and all enrolled NL speakers as a single distribution, which helps calibrate test scores and select a consistent threshold over the entire NL. LSH is applied to find NL speakers as well as utterances in the normalization cohort that are most similar to a test utterance, so that PLDA scoring is performed only on a small subsets of utterances, which significantly lowers the computation cost and latency of the NL detection. The effectiveness of LSH is further amplified when using different adaptive lengths for Z-Norm and T-Norm terms in AS-Norm and NL-Norm, so that evaluating a relatively small number of similarity scores between a test utterance and the normalization cohort is sufficient to reach optimal accuracy.

While the experiments are conducted on i-vectors, none of the techniques proposed in this paper is specific to i-vector, therefore we expect these approaches can be applied to x-vectors as well, which will be the subject of a future study, along with more direct comparisons with results from other NL detection methods.

References

Roland Auckenthaler, Michael Carey, and Harvey Lloyd-Thomas. 2000. Score normalization for text-independent speaker verification systems. *Digital Signal Processing*, 10(1):42–54.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proc. of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 380–388. ACM.

Sandro Cumani, Pier Batzu, Daniele Colibro, Claudio Vair, Pietro Laface, and Vasileios Vasilakakis. 2011. Comparison of speaker recognition approaches for real applications. In *Proc. of the Annual Conf. of the Intl. Speech Communication Association, INTERSPEECH*, pages 2365–2368.

Najim Dehak, Patrick J. Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. 2011. Front-end factor analysis for speaker verification. *IEEE Trans. on Audio, Speech, and Language Processing*, 19(4):788–798.

Roberto Font. 2019. A denoising autoencoder for speaker recognition. results on the mce 2018 challenge. In *ICASSP 2019 - 2019 IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6016–6020.

J. Fortuna, P. Sivakumaran, A. Ariyaeeinia, and A. Malegaonkar. 2005. Open-set speaker identification using adapted Gaussian mixture models. In *Proc. Interspeech 2005*, pages 1997–2000.

J. Fortuna, P. Sivakumaran, A. M. Ariyaeeinia, and A. Malegaonkar. 2004. Relative effectiveness of score normalisation methods in open-set speaker identification. In *Proc. The Speaker and Language Recognition Workshop (Odyssey)*, pages 369–376.

Craig S. Greenberg, Lisa P. Mason, Seyed Omid Sadjadi, and Douglas A. Reynolds. 2020. Two decades of speaker recognition evaluation at the national institute of standards and technology. *Computer Speech & Language*, 60:101032.

Nancie Gunson, Diarmid Marshall, and Mervyn Jack. 2015. Effective speaker spotting for watch-list detection of fraudsters in telephone banking. *IET Biometrics*, 4(2):127–136.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. of the 30th Annual ACM Symposium on Theory of Computing*, STOC '98, page 604–613, New York, NY, USA. ACM.

Woojay Jeon and Yan-Ming Cheng. 2012. Efficient speaker search over large populations using kernelized locality-sensitive hashing. In *2012 IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4261–4264.

Zahi Karam, William Campbell, and Najim Dehak. 2011. Towards reduced false-alarms using cohorts. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 Intl. Conf. on*, pages 4512–4515.

Patrick Kenny. 2010. Bayesian Speaker Verification with Heavy-Tailed Priors. In *Proc. The Speaker and Language Recognition Workshop (Odyssey 2010)*, page paper 14.

Elie Khoury, Khaled Lakhdhar, Andrew Vaughan, Ganesh Sivaraman, and Parav Nagarsheth. 2019. Pindrop Labs' Submission to the First Multi-Target Speaker Detection and Identification Challenge. In *Proc. Interspeech 2019*, pages 1502–1505.

Ryan Leary and Walter Andrews. 2014. Random projections for large-scale speaker search. In *Proc. Interspeech 2014*, pages 66–70.

K. Li and J. Porter. 1988. Normalizations and selection of speech segments for speaker recognition scoring. In *ICASSP-88., Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 595–598.

Jeff Ma, Jan Silovsky, Man-hung Siu, and Owen Kimball. 2015. Large-scale speaker search using plda on mismatched conditions. In *2015 IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1846–1850.

Amit Malegaonkar and Aladdin Ariyaeeinia. 2011. Performance evaluation in open-set speaker identification. In *Biometrics and ID Management*, pages 106–112, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pavel Matejka, Ondrej Glembek, Fabio Castaldo, Md Jahangir Alam, Oldrich Plchot, Patrick Kenny, Lukas Burget, and Jan Cernocký. 2011. Full-covariance ubm and heavy-tailed plda in i-vector speaker verification. In *ICASSP, IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, pages 4828–4831.

Pavel Matějka, Ondřej Novotný, Oldřich Plchot, Lukáš Burget, Mireia Diez Sánchez, and Jan Černocký. 2017. Analysis of Score Normalization in Multilingual Speaker Recognition. In *Proc. Interspeech 2017*, pages 1567–1571.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Vesel. 2011. The Kaldi speech recognition toolkit. *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*.

Simon J.D. Prince and James H. Elder. 2007. Probabilistic linear discriminant analysis for inferences about identity. In *2007 IEEE 11th Intl. Conf. on Computer Vision*, pages 1–8.

Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. 2000. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10(1):19–41.

Seyed Omid Sadjadi, Craig Greenberg, Elliot Singer, Douglas Reynolds, Lisa Mason, and Jaime Hernandez-Cordero. 2020. The 2019 NIST Speaker Recognition Evaluation CTS Challenge. In *Proc. The Speaker and Language Recognition Workshop (Odyssey)*, pages 266–272.

Ludwig Schmidt, Matthew Sharifi, and Ignacio Lopez Moreno. 2014. Large-scale speaker identification. In *2014 IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1650–1654.

Suwon Shon, Najim Dehak, Douglas Reynolds, and James Glass. 2019. MCE 2018: The 1st Multi-Target Speaker Detection and Identification Challenge Evaluation. In *Proc. Interspeech 2019*, pages 356–360.

Suwon Shon, Younggun Lee, and Taesu Kim. 2018. Large-scale speaker retrieval on random speaker variability subspace. *arXiv preprint arXiv:1811.10812*.

Elliot Singer and Douglas A. Reynolds. 2004. Analysis of multitarget detection for speaker and language recognition. In *Proc. The Speaker and Language Recognition Workshop (Odyssey)*, pages 301–308.

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. 2018. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333.

Kevin Wilkinghoff. 2020. On Open-Set Speaker Identification with I-Vectors. In *Proc. The Speaker and Language Recognition Workshop*, pages 408–414.

Yaniv Zigel and Moshe Wasserblat. 2006. How to deal with multiple-targets in speaker identification systems? In *2006 IEEE Odyssey - The Speaker and Language Recognition Workshop*, pages 1–7.

Supervised Acoustic Embeddings And Their Transferability Across Languages

Sreepratha Ram
UAE University
sree_ram@uaeu.ac.ae

Hanan Aldarmaki
MBZUAI
hanan.aldarmaki@mbzuai.ac.ae

Abstract

In speech recognition, it is essential to model the phonetic content of the input signal while discarding irrelevant factors such as speaker variations and noise, which is challenging in low-resource settings. Self-supervised pre-training has been proposed as a way to improve both supervised and unsupervised speech recognition, including frame-level feature representations and Acoustic Word Embeddings (AWE) for variable-length segments. However, self-supervised models alone cannot learn perfect separation of the linguistic content as they are trained to optimize indirect objectives. In this work, we experiment with different pre-trained self-supervised features as input to AWE models and show that they work best within a supervised framework. Models trained on English can be transferred to other languages with no adaptation and outperform self-supervised models trained solely on the target languages.

***Keywords*—** Unsupervised ASR, Transfer Learning, Acoustic Word Embeddings

1 Introduction

With supervised speech recognition systems getting more robust and accurate due to the availability of large amounts of labeled data and computational power (Gulati et al., 2020; Baevski et al., 2020b), more attention is now given to low-resource languages for which training data are limited or non-existent (Aldarmaki et al., 2022). Unsupervised pre-training using unlabeled speech can be leveraged to improve both supervised and unsupervised models; for instance, speech representations pre-trained on large amounts of unlabeled speech from multiple languages have been shown to improve ASR performance for low-resource languages (Kawakami et al., 2020; Conneau et al., 2020).

While most supervised ASR models operate at the level of phones, word-level segmental ASR where variable-length segments are modeled and embedded into fixed-dimensional vectors have also been explored with relative success (Abdel-Hamid et al., 2013; He and Fosler-Lussier, 2015). In a similar vein, Acoustic Word Embeddings (AWEs) have been proposed as a way to efficiently compare variable-length speech segments in low-resource settings (Peng et al., 2020; Kamper et al., 2020). Unlike written words, spoken words naturally contain speaker and phonetic variability that makes them more difficult to model in a latent space without supervision. Self-supervised pre-training and cross-lingual transfer are two possible approaches to make unsupervised models more robust to non-linguistic variations in the input signal.

In this work, we investigate the performance of self-supervised training of AWE models versus supervised training with zero-shot cross-lingual transfer. We experiment with different types of acoustic features and measure their performance separately and within the AWE models. While we find that pre-trained acoustic features improve the performance of self-supervised AWE models to some extent, a larger improvement can be achieved when the AWE models are trained in a supervised manner using small amount of labeled data from a different language. This zero-shot cross-lingual transfer is observed consistently across different languages, and particularly with the use of pre-trained feature representations. Our results suggest that supervised training with zero-shot cross-lingual transfer is a more effective approach for low-resource speech models compared with purely self-supervised training[1].

2 Background & Related Work

Spoken language is often modeled using short fixed-length frames of 10 to 30 ms duration, which

[1]We provide python training and evaluation scripts for replicating our experiments: https://github.com/h-aldarmaki/acoustic_embeddings

results in variable-length word segments. Dynamic Time Warping (DTW) is an early technique that uses dynamic programming to compare variable-length segments by finding optimal frame-wise alignment. DTW is rather inefficient, which motivates embedding variable-length segments into vectors of fixed size that can be compared using more efficient metrics such as cosine or Euclidean distance (Levin et al., 2013). Different types of Acoustic Word Embeddings (AWE) have been proposed. As these techniques are generally meant for low-resource languages, they are typically trained in a self-supervised manner, most commonly using an auto-encoder network with reconstruction loss (Chung et al., 2016; Holzenberger et al., 2018). Compared with direct comparison via DTW, these AWEs generally result in similar or slightly superior performance while being far more efficient (Holzenberger et al., 2018). Peng et al. (2020) describes an alternative training strategy using correspondence auto-encoders, which relies on word pairs extracted via unsupervised spoken term discovery, and further improvements can be achieved using contrastive learning and multi-lingual adaptation (Jacobs et al., 2021).

The above models use static acoustic features (e.g. MFCCs) as input. van Staden and Kamper (2021) shows that using pre-trained features like CPC (van den Oord et al., 2018) improves the performance of unsupervised AWE models. Pre-trained features have been repeatedly shown to improve performance in supervised downstream tasks (Yang et al., 2021). In addition, pre-trained features have been shown to transfer across languages. For instance, a modified version of CPC (MCPC) is described in Riviere et al. (2020), which demonstrates that pre-training these features on Egnlish results in improved phone classification accuracy for other languages. Other types of pre-trained features, such as wav2vec 2.0 (Baevski et al., 2020a) have been shown to improve both supervised and unsupervised ASR performance (Baevski et al., 2021), and multi-lingual training of these features (i.e. XLSR-53) can lead to improvements across many languages compared to monolingual pre-training (Conneau et al., 2020).

3 Objectives & Methodology

The objective of this study is to investigate the effectiveness and trasnsferability of pre-trained acoustic features when used as input to acoustic word embeddings. To that end, we compare self-supervised AWEs trained directly on the target languages versus zero-shot cross-lingual transfer of supervised AWEs trained on a different source language. To our knowledge, the combination of pre-trained features with AWE models has not been fully investigated; most AWE models are trained with standard acoustic features like MFCCs, while self-supervised features are typically evaluated within supervised models fine-tuned for the target languages. Furthermore, zero-shot cross-lingual transfer of supervised AWEs has not been the focus of previous works in this area, which mainly focused on improving self-supervised AWEs.

For the purpose of this evaluation, we use a relatively simple architecture for the embedding model and we fix the hyper-parameters based on preliminary validation results for English self-supervised AWEs[2]. We do not do any further tuning of the self-supervised or the supervised models. We use English as the source language, and evaluate zero-shot transfer on four other languages: French, German, Spanish, and Arabic, with the latter used as a challenge set since it contains more variability and noise. No labeled data were used for the target languages with the exception of word boundaries which were obtained via force alignment. We evaluate mainly using minimal-pair ABX error rates to measure phonetic discriminability and speaker invariance. We also cluster the embedded words and measure how often different occurrences of the same words end up in the same cluster.

4 Experimental Settings

4.1 Model Architecture

Our AWE model consists of a multi-layer bidirectional LSTM encoder, followed by a uni-directional LSTM decoder, similar to Chung et al. (2016) and (Holzenberger et al., 2018). The encoder takes a sequence of T acoustic features representing one spoken word. The forward and backward states of the last hidden layer of the encoder are concatenated and used as an embedding of the given word, call it $\mathbf{h}^T$. The decoder generates the target sequence one step at a time, conditioned on $\mathbf{h}^T$ and the output at the previous time step, similar to Chung and

[2]We observed that self-supervised models were very sensitive to the choice of architecture and hyper-parameters, so we fixed these in favor of self-supervised models. As shown in later sections, we still got better results with the supervised models, which shows that they are more robust and easier to optimize on top of being more effective.

Glass (2018). In the self-supervised setting, the target sequence is the same as the input sequence, so the model is trained as an auto-encoder with MSE loss. In the supervised setting, the target is a sequence of phonemes representing the input word, and the model is trained by minimizing the negative log-likelihood. We used 2-layer networks with 100 hidden units for most models, which results in embeddings of size 200. We also used dropout with probability 0.3 on the input features, similar to the denoising networks used in Chung et al. (2016). More details of the parameters and training process can be found in the Appendix.

4.2 Feature Extraction

For easier reproduciblity, we used the s3prl toolkit[3] for extracting all features. We used the pre-trained s3prl upstream models; among the many pretrained self supervised speech representations available, modified CPC, Wav2Wec2 and XLSR-53 were chosen based on superior DTW-based ABX scores[4]. All pre-trained models, with the exception of XLSR, have been exclusively pre-trained on English data. XLSR-53 was pre-trained on unlabeled speech from 53 languages, including all target languages in our experiments. As observed by other researchers (Bartelds et al., 2022), the performance of features extracted from transformer-based models is largely dependent on the choice of layer; we used the last hidden layer for modified CPC, the second to last hidden layer for Wav2Vec2 and the central hidden layer (layer 12) for XLSR-53. Averaging all layers gave reasonable results, but these choices led to the best performance. For MFCC features, we also used the s3prl implementation, which includes 13 static features as well as dynamic delta and delta-delta coefficients.

4.3 Data

We used the Librispeech (Panayotov et al., 2015) and Multilingual Librispeech (Pratap et al., 2020) datasets for English (en), French (fr), German (de), and Spanish (es). We used the dev sets for training, and test sets for evaluation (dev-clean and test-clean for English). We obtained the word boundaries automatically by forced alignment. For Arabic (ar), we used the dev and test sets of MGB2 (Ali et al., 2016). This dataset is expected to be more challenging as it contains a diversity of dialects as well as various noise conditions. See the Appendix for more details on the datasets and the word alignment process.

4.4 Evaluation Scheme

We constructed Minimal-Pair ABX tasks, as described in Schatz et al. (2013). ABX tasks are typically used to measure phoneme discrimination in zero-resource settings, and they consist of two segments, A and B, that differ by a minimal contrast (e.g. one phoneme difference), and a third segment X that matches either A or B. A distance measure such as DTW or cosine is used to find the closest match. We used two variants of this task: within-speaker ABX, where all three words are spoken by the same speaker, and cross-speaker ABX, where X is spoken by a different speaker. We automatically extracted the words from each test set; we selected A and B by finding word pairs that have the same length[5] and Levenshtein edit distance of 1 or 2, which roughly corresponds to a difference of one or two phonemes most of the time. For Arabic, the dataset did not have speaker ids, so all three words could be from different speakers. In addition, due to the lower quality of the sound recordings and the presence of noise in this dataset, the word alignment quality is much lower than the other languages, so the automatic process resulted in many invalid segments. To have a more reliable test set for Arabic, we manually checked the validity of the extracted words and kept 954 validated word pairs for evaluation.

We also used clustering for complementary evaluation. We clustered the embeddings using K-Means with K being the number of unique words in the test set. We calculated the accuracy of clustering as the percentage of words that match their cluster label, which is the word id of the majority of segments in each cluster. This allows us to measure if the embeddings of the same words are similar enough to be clustered together.

5 Results

Table 1 shows ABX error rates using the input features directly (with DTW as distance metric), self-supervised AWEs trained on each language,

[3] https://github.com/s3prl/s3prl

[4] We did experiment with other features like APC, VQ-APC, and VQ-Wav2Vec, and got similar or inferior performance to MCPC and Wav2Vec2. We opted to omit these for brevity.

[5] Since automatic word alignments tend to be inaccurate around the boundaries, we only used words that have at least five characters.

	en		fr		de		es		ar
	within	across	within	across	within	across	within	across	across
Using DTW									
MFCC	9.98	19.85	12.59	24.82	11.46	25.03	11.83	25.27	40.98
wav2vec	8.51	11.15	9.95	15.61	9.01	15.08	10.13	15.46	37.42
MCPC	7.80	11.74	9.96	17.09	9.22	15.85	11.14	18.03	38.99
XLSR-53	9.45	13.72	10.97	16.77	10.89	15.76	14.31	19.31	40.15
Self-Supervised AWEs in each language									
MFCC	12.30	19.12	16.63	25.06	16.71	25.99	16.58	25.21	43.92
wav2vec	6.63	9.27	9.94	13.19	10.56	15.09	12.25	15.23	38.16
MCPC	7.66	9.53	11.64	16.19	10.24	15.30	13.25	16.29	41.09
XLSR-53	10.61	12.19	13.72	16.19	12.59	15.73	17.10	20.14	37.00
Supervised AWEs trained on English									
MFCC	3.83	4.57	10.77	15.32	9.16	13.06	11.49	16.56	38.15
wav2vec	1.38	1.14	6.59	9.44	4.98	7.32	7.32	10.12	34.80
MCPC	2.49	2.51	8.13	12.23	6.66	10.68	9.89	13.99	39.20
XLSR-53	**0.93**	**0.79**	**4.12**	**5.71**	**1.92**	**2.83**	**5.05**	**6.21**	**31.76**

Table 1: ABX error rates (%) within and across speakers for each language

	en	fr	de	es	ar
Self-Supervised AWEs for each language					
MFCC	47.3	48.4	45.0	54.3	30.9
wav2vec	66.1	59.9	59.5	67.5	35.9
MCPC	57.2	53.6	53.9	60.4	33.5
XLSR-53	52.0	52.2	54.9	55.0	31.0
Supervised AWEs trained on English					
MFCC	68.5	52.7	56.7	61.1	33.4
wav2vec	82.3	64.8	69.3	71.1	38.8
MCPC	74.5	56.4	62.7	66.2	35.6
XLSR-53	**84.3**	**69.1**	**78.1**	**75.8**	**41.8**

Table 2: K-Means Clustering Accuracy (%)

and supervised AWEs trained on English. Cosine similarity is the metric used in the latter two settings. Confirming previous results (Riviere et al., 2020), we do observe that pre-trained acoustic features like Modified CPC and Wav2Vec2, which are trained exclusively on English unlabeled speech, transfer well across languages. These pre-trained features consistently outperformed MFCC features for all languages, particularly in cross-speaker evaluation. Unsurprisingly, the English language has the best ABX scores overall simply because the pre-trained features used are all trained on English. The results for self-supervised AWE models are mixed, but generally they are in the same range as DTW performance, which also conforms with previously published results (Holzenberger et al., 2018).

With supervised training, we see significant reduction in errors rates for all languages. The lowest error rates are achieved on the English test set, as expected. More notably, the largest reduction in error rates is achieved with the XLSR features. It is also interesting to note that XLSR features were not impressive in the self-supervised setting compared with other features; Wav2Vec2 and MCPC, which were trained on English only, gave better results in the self-supervised framework for all test languages. The advantage of using these cross-lingual features was only evident in the supervised and transfer learning setting, where they consistently outperformed all other features. For Arabic, the error rates are higher overall due to the nature of the dataset, but we still observe the lowest error rate in the transfer learning setting.

Finally, we see in table 2 that the clustering accuracy results are consistent with the ABX results, where supervised models trained on English consistently gave higher accuracy compared with self-supervised models trained on the target languages.

6 Conclusions

Our results demonstrate the superior effectiveness of zero-shot transfer learning of acoustic word embeddings compared with self-supervised training in the target languages. This is particularly useful for low-resource languages for which data may not be available for supervised or self-supervised

training. The mechanism of this transfer is mainly through the reduction in speaker variability which is far easier to achieve via supervised training. In addition, supervised training makes the most out of pre-trained features, where we see further reduction in error rates that far exceed the reduction observed in self-supervised settings. The presence of noise naturally results in larger error rates; further investigations are needed to demonstrate the transferability of noise robustness in a similar manner.

Acknowledgement

This work was supported by grant no. 31T139 at United Arab Emirates University and partially funded under UAEU-ZU Joint Research Grant G00003715 (Fund No.: 12T034) through Emirates Center for Mobility Research.

References

Ossama Abdel-Hamid, Li Deng, Dong Yu, and Hui Jiang. 2013. Deep segmental neural networks for speech recognition.

Hanan Aldarmaki, Asad Ullah, Sreepratha Ram, and Nazar Zaki. 2022. Unsupervised automatic speech recognition: A review. *Speech Communication*.

Ahmed Ali, Peter Bell, James Glass, Yacine Messaoui, Hamdy Mubarak, Steve Renals, and Yifan Zhang. 2016. The mgb-2 challenge: Arabic multi-dialect broadcast media recognition. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 279–284. IEEE.

Alexei Baevski, Wei-Ning Hsu, Alexis Conneau, and Michael Auli. 2021. Unsupervised speech recognition. *arXiv preprint arXiv:2105.11084*.

Alexei Baevski, Henry Zhou, Abdel rahman Mohamed, and Michael Auli. 2020a. wav2vec 2.0: A framework for self-supervised learning of speech representations. *ArXiv*, abs/2006.11477.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020b. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460.

Martijn Bartelds, Wietse de Vries, Faraz Sanal, Caitlin Richter, Mark Liberman, and Martijn Wieling. 2022. Neural representations for modeling variation in speech. *Journal of Phonetics*, 92:101137.

Mathieu Bernard and Hadrien Titeux. 2021. Phonemizer: Text to phones transcription for multiple languages in python. *Journal of Open Source Software*, 6(68):3958.

Yu-An Chung and James Glass. 2018. Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech. *Proc. Interspeech 2018*, pages 811–815.

Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. *Interspeech 2016*, pages 765–769.

Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. 2020. Unsupervised cross-lingual representation learning for speech recognition. *arXiv preprint arXiv:2006.13979*.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. *Proc. Interspeech 2020*, pages 5036–5040.

Yanzhang He and Eric Fosler-Lussier. 2015. Segmental conditional random fields with deep neural networks as acoustic models for first-pass word recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Nils Holzenberger, Mingxing Du, Julien Karadayi, Rachid Riad, and Emmanuel Dupoux. 2018. Learning word embeddings: Unsupervised methods for fixed-size representations of variable-length speech segments. *Proc. Interspeech 2018*, pages 2683–2687.

Christiaan Jacobs, Yevgen Matusevych, and Herman Kamper. 2021. Acoustic word embeddings for zero-resource languages using self-supervised contrastive learning and multilingual adaptation. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 919–926. IEEE.

Herman Kamper, Yevgen Matusevych, and Sharon Goldwater. 2020. Multilingual acoustic word embedding models for processing zero-resource languages. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6414–6418. IEEE.

Kazuya Kawakami, Luyu Wang, Chris Dyer, Phil Blunsom, and Aaron van den Oord. 2020. Learning robust and multilingual speech representations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1182–1192.

Keith Levin, Katharine Henry, Aren Jansen, and Karen Livescu. 2013. Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 410–415. IEEE.

Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. 2017.

Montreal forced aligner: Trainable text-speech alignment using kaldi. *Proc. Interspeech 2017*, pages 498–502.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE.

Puyuan Peng, Herman Kamper, and Karen Livescu. 2020. A correspondence variational autoencoder for unsupervised acoustic word embeddings. *Advances in neural information processing systems*.

Vineel Pratap, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve, and Ronan Collobert. 2020. MLS: A large-scale multilingual dataset for speech research.

Morgane Riviere, Armand Joulin, Pierre-Emmanuel Mazaré, and Emmanuel Dupoux. 2020. Unsupervised pretraining transfers well across languages. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7414–7418. IEEE.

Thomas Schatz, Vijayaditya Peddinti, Francis Bach, Aren Jansen, Hynek Hermansky, and Emmanuel Dupoux. 2013. Evaluating speech features with the minimal-pair abx task: Analysis of the classical mfc/plp pipeline. In *INTERSPEECH 2013: 14th Annual Conference of the International Speech Communication Association*, pages 1–5.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv e-prints*, pages arXiv–1807.

Lisa van Staden and Herman Kamper. 2021. A comparison of self-supervised speech representations as input features for unsupervised acoustic word embeddings. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 927–934. IEEE.

Shu-wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y Lin, Andy T Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, et al. 2021. Superb: Speech processing universal performance benchmark. *arXiv preprint arXiv:2105.01051*.

A Appendix

A.1 Dataset Details

A.2 Model Architecture & Hyper-Parameters

The architecture described in section 4.1 was modeled after other acoustic word embedding models (Chung et al., 2016; Chung and Glass, 2018; Holzenberger et al., 2018) with slight variations in details. We found that this particular configuration worked best across different acoustic features,

Dataset	test	dev
English	52,576	54,402
French	90,958	83,560
German	121,713	122,903
Spanish	88,417	87,417
Arabic	62,745	57,532

Table 3: Total number of words in each dataset

whereas other choices gave mixed results. For example, using GRUs instead of LSTMs worked well with pre-trained features but was worse for MFCCs. The decoding process described in Holzenberger et al. (2018), where positional encodings are used instead of previous outputs also resulted in inferior performance. We also found that using teacher forcing instead of the model's previous output as input to the decoder hurt the performance. Finally, using two layers was crucial to get results in line with DTW perforamnce for most self-supervised models. The only exception is the self-supervised model with XLSR features which resulted in unstable training with 2 layers. We found it to work much better with a single layer network and slightly larger embedding size. Generally, larger embeddings sizes improved performance to some extent, but the improvements were smaller beyond the values that we have chosen; furthermore, using smaller sizes is more advantageous in terms of computational efficiency. We did not perform any hyper-parameter tuning for the target languages since we are working within the premise of low-resource settings where validation data may not be available.

Table 4 shows the number of parameters for each model. Since the decoder is only used for training and can be discarded after that, we only show the number of encoder parameters.

A.3 Training Details

The supervised models were trained with NLL loss, and the training targets are sequences of phonemes obtained using the Phonemizer package [6] (Bernard and Titeux, 2021). This choice seemed more sensible at first, but we found that using sequences of characters instead of phonemes worked equally well.

The model was implemented using PyTorch and trained on NVIDIA K80 GPU as provided in AWS p2.xlarge instances. For optimization, we found

[6] https://github.com/bootphon/phonemizer

Model	input	hidden	no.of parameters
Self-Supervised			
MFCC	39	100	354,400
Wav2Vec	768	100	937,600
MCPC	256	100	528,000
XLSR-53	1024	250	1,411,200
Supervised			
MFCC	39	100	354,400
Wav2Vec	768	100	937,600
MCPC	256	100	528,000
XLSR-53	1024	100	1,142,400

Table 4: Input size, hidden layer size, and total number of encoder parameters for each model.

that adam optimizer worked for all features except MFCCs, for which SGD with cyclical or step learning rate schedule was more stable.

Table 3 shows the number of words in each dataset. The word alignments were obtained via force alignment using The Montreal Forced Aligner[7] (McAuliffe et al., 2017) for English, German, French, and Spanish. The Montreal aligner uses an ASR engine, and since these datasets are relatively clean, the alignments are generally accurate. For Arabic, the best option was the aeneas toolkit[8], which relies on a TTS engine to align the synthesized words with the actual audio segments. We used Amazon Polly TTS for higher quality, but overall the alignments were not as accurate as the other datasets, which we believe is due to the low quality of the recordings, presence of noise, and high variability in accents. The low clustering accuracy could be partially attributed to the inaccurate labeling of the segments as a result of this. For ABX evaluation on the Arabic set, we manually filtered the segments that had somewhat accurate boundaries; the chosen pairs still contained high level of noise conditions, such as background music and interfering speech.

[7] https://github.com/MontrealCorpusTools/Montreal-Forced-Aligner

[8] www.readbeyond.it/aeneas

A Dataset for Detecting Humor in Arabic Text

Hend Al-Khalifa*, Fetoun AlZahrani, Hala Qawara, Reema AlRowais, Sawsan Alowa and Luluh AlDhubayi
Information Technology Department, College of Computer and Information Sciences
King Saud University
Riyadh, Saudi Arabia
*Hendk@ksu.edu.sa

Abstract

Humor detection is a complex and ambiguous task in natural language processing. This has made automatic humor detection challenging, particularly for languages with limited resources such as Arabic. In this paper, we attempt to solve this task by collecting and annotating Arabic humorous tweets in dialects and Modern Standard Arabic (MSA) text then performing automatic humor detection on the collected data. We experimented on the collected dataset by fine-tuning seven Arabic Pre-Trained language models (PLMs) which are: AraBERTv02, Arabertv02-twitter, QARIB, MarBERT, MARBERTv2, CAMeLBERT-DA, and CAMeLBERT-MIX to establish a baseline classification system. We concluded that CAMeLBERT-DA was the best-performing model and it achieved an F1-score and accuracy of 72.11%.

1 Introduction

Humor is defined as the attempt of practices to provoke laughter and provide a sort of amusement. Humor permeates human life in both dimensions' reality and virtually (Zhang and Liu, 2014). Humor has become a frequent subject of research due to its coercive power in communication and other fields (Meyer, 2000). For instance, computational and linguistic research in humor have been going for more than twenty years, and according to (Amin and Burghardt, 2020), they mainly focus on Humor Recognition, Humor Adoption Systems, Computational Humor Evaluation, Computational Humor Applications and Computational Humor Datasets and Corpora. In fact, the power of social media and the popularity of humor in it encouraged some Natural Language Processing (NLP) researchers to conduct their studies about humor on Twitter and other social media platforms such as (Zhang and Liu, 2014), (Khandelwal et al. 2018) and (Meaney et al. 2021). Drawing on the humor context, Arabic linguistic research on humor is sparse, some of it focused on certain Arabic dialects and humor aspects e.g. (Nayef and El-Nashar, 2014), (Omar, 2017), (Banikalef, 2014) and (Bzour, 2021) conducted an analysis of Arabic humor.

According to (Altin 2019), the main benefit of detecting humor is to improve human machine communication, as the machines are not fully capable to understand humor as humans do. Therefore, detecting humor is very essential to make progression in human alike and intelligent systems.

The lack of high quality annotated data for conducting various experiments is one of the main obstacles to developing any detection models in low-resource languages. In order to address this issue, this study is the first to our knowledge to detect humor in text written in Arabic language, whether it is MSA or Arabic Dialect. The study includes details about the process of collecting, preprocessing, analyzing the data, and the encountered challenges during the process. The dataset we collected is publicly available[1].

The rest of the paper is organized as follows: Section 2 presents previous work in the area of humor detection. Section 3, provides detailed description of the dataset. Section 4, briefly describes the used experiments and analysis of results. Finally, Section 5 concludes the paper with future work.

2 Related work

The task of detecting humor in text has drawn the attention of many researchers in different languages. Although there is no single definition of

[1] https://github.com/iwan-rg/Arabic-Humor

humor, most systems focus on identifying jokes, irony, and other forms of verbal play. Early humor detection systems were based on rule-based approaches, which relied on hand-crafted rules to identify humor. However, these systems were limited in their ability to generalize to new types of humor (Rajakumar et al., 2010).

More recent systems have employed machine learning techniques to automatically learn rules for identifying humor. These approaches have shown promise, but still face challenges in accurately detecting humor. For instance, Hossain et al.(2019) analysed regular English news headlines to predict whether or not an edited headline is funny. Kao et al. (2016) did a humor detection in puns in English text at a fine-grained level. Similarly, a novel annotation scheme in Chinese text contained annotated humorous text and keywords that triggered humor, with 9,123 Chinese jokes and 39,977 sentences in total (Zhang et al. 2019). HEMOS (Humor-EMOji-Slang-Based) is a deep learning system for sentiment classification of Chinese language in two collected lexicons (slang expression and converted Weibo[2] emojis). It has a binary annotation (optimistic humorous and pessimistic humorous) which added to the standard positive and negative sentiments. The carried experiment was implemented on both lexicons to an attention-based bi-directional long short-term memory recurrent neural network (AttBiLSTM). It resulted in a substantial improvement in predicting sentiment polarity on Weibo (Li et al. 2020). Furthermore, a Spanish based task by (Castro et al. 2016) to detect humor on a crowdsourced corpus of labelled data classified by implementing number of features such as adult slang, dialogue, hashtag, keywords, etc. Likewise, a neural network for humor recognition using (biLSTM) models used to classify tweets in Spanish as humorous or not. Additionally, it scored the level of funniness in the context based on Human annotation from one to five where one (not funny) and five (excellent) (Altin et al. 2019).

To tackle the problem of humor detection the attention of research has shifted to deep learning approaches as seen in the two previous studies and as in (Annamoradnejad and Zoghi 2020), who used sentence embeddings and utilized the linguistic structure of humor in designing their proposed model. Their model outperforms the baseline models and achieved high accuracy in detecting humor in their ColBERT experiment on labelled dataset, which is technically injecting BERT sentence embedding into a neural network model that processes sentences separately in parallel hidden layers. Also, Fan et al.(2020) proposed an internal and external attention neural network (IEANN) for the humor detection task by merging two types of attention mechanisms to capture the incongruity and ambiguity in humorous text. An experiment conducted on two humor datasets to test the proposed model and the results showed that their model has better interpretability as they claimed. Researchers have shown an increased interest in automatic humor detection. A data collected from Twitter and Kaggle[3] was evaluated using several models to the purpose of detecting and rating humor and offensive language, interestingly, they also originally predict humor controversy which result from the variance of annotators rating of certain jokes (Meaney et al. 2021).

Other than text-based detection of humor, multimodal language dataset were used in (Hasan et al. 2019) and (Bertero and Fung, 2016). The first one, determined the effects of using (text, vision and audio) all in one dataset called UR-FUNNY for humor detection task and present the performance results of the task. The latter proposed a deep neural network framework of integration of audio and language features for their dataset that collected from three TV shows that have canned laughter which used as indication of the humor occurrence.

Finally, related works for humor detection in Arabic language is scarce. They are usually considered as part of sarcasm detection task, therefore, we aim in this study to build the first Arabic humor dataset and evaluate it using different Arabic PLMs, to establish a baseline classification system for this downstream task.

3 Arabic Humor Dataset

3.1 Data Collection

To collect humorous text in Arabic language different tools were used. Twint [4] and Sketch Engine[5] are used to create a corpus of more than 10,000 entries. Firstly, tweets were scrapped from Twitter using Twint by following two approaches.

[2] is the largest Chinese social network.
[3] https://www.kaggle.com/
[4] https://github.com/twintproject/twint
[5] https://www.sketchengine.eu/

The first approach, scrapping using username where multiple accounts known for their humorous content were selected. The second approach, scrapping using trending hashtags such as "نفسيه"and "نكت" which are related to humor aspect. The list of used usernames and hashtags are given in Table 1.

Secondly, Sketch Engine was used for scrapping humorous text by using the web search tool that was provided by the Sketch Engine. By entering a set of keywords that are likely used for triggering humor to collect data that are relatively contain humorous text. The list of used keywords is given in Table 1.

Twitter		**Web Search**
Usernames	**Hashtags**	**Keywords**
7amadQalam, _3urf, Fake_SteveJobs1 iiim7mdz, mnm2900, Jokes_HD, Jokes_LoooL, nektah_time	#وليد_مع_اكشن #نفسيه	مرة محشش، في محشش، محشش، مرة واحد، في عجوز، غبي، مصري، حضرمي، حضرميه، فيه واحد، واحد تزوج، فيه سوداني، فيه مجنون، قصيمي، كان فيه قصيمي، جحا، فيه صعيدي، حوطي، حواطى، فيه جنوبي، جيزاني، غمد وزهارين، مسطول، منسم

Table 1: Lists of Usernames, Hashtags, and Keywords

3.2 Data Cleaning and Preprocessing

The collected text for the dataset are unstructured with slang terms, abbreviations, and orthographic errors that should be cleaned and pre-processed before performing the corpus analysis properly. Thus, the unstructured text must be converted to a structured format using pre-processing techniques, which are an important step in natural language processing (Aljabri et al. 2021).

- Arabic normalization:
  The unification of some characters that have many forms, was employed on the dataset by replacing (ى،ئ) with (ي), letter (ؤ) with (و), letter (ة) to (أ،إ،آ) to (ا) and (گ) to (ك).
- Deleted unrelated and unwanted text:
  The extracted data from the web was not limited to certain tags; it extracted all the content from the web pages that match the search keywords. As a result, the data contained unrelated information such as information about the topic writer, number of contributions and contact information; hence most of the collected web data were from forums.

Moreover, the data cleaning steps that were applied on web data were as follow:

- Remove unrelated information:
  A dictionary of unrelated words was created. The dictionary contains the following keywords: " عضو"، "المحتويات"، "كاتب الموضوع"، "عدد المساهمات"، "عدد الرسائل"، "معلومات الاتصال"،"بيانات العضو"،"مساهمة". After that a special query is created to search through the data frame to drop any row that contains one or more of the keywords in the dictionary.
- Remove diacritics or Arabic Tashkeel:
  Diacritics is a vowel mark used to indicate short vowels which are: fatha, damma, and kasra, in addition to sukun and shadda which indicates the absence of a vowel. Diacritics are removed to reduce data sparsity.
- Remove numbers, English letters, special characters, symbols, emojis, and newlines.
- Remove all words that contain underscore and hashtag sign from the data.
- Remove repeated characters such as "هههه".
- Remove duplicate rows from the dataset.

Finally, after applying the preprocessing and cleaning of the data, we performed tokenization where we split the text into small pieces called tokens. A simple tokenizer was implemented which is space-based. This tokenizer splits the text into chunks by its spaces using regular expression (regex). Table 2 demonstrated an example from the dataset and how it changed during the data preprocessing and cleaning process.

Process	**Change**
Raw Tweet	محشش حط على السيارة برقع ليييييييش علشانها نكت# 😅👏💔 باسم امه
Normalization	محشش حط علي السياره برقع ليييييييش علشانها نكت# 😅👏💔 باسم امه
Remove Hashtags	محشش حط علي السياره برقع ليييييييش علشانها 😅👏💔 باسم امه
Remove Emojis	محشش حط علي السياره برقع ليييييش علشانها باسم امه
Remove Repeating Char	محشش حط علي السياره برقع ليش علشانها باسم امه
Simple Tokenization using regex (space-based)	['محشش'، 'حط'، 'علي'، 'السياره'، 'برقع'، 'ليش'، 'علشانها'، 'باسم'، 'امه"]

Table 2: Pre-processing and Data Cleaning in Tweet.

3.3 Dataset Description

The purpose of the collected dataset is to detect humor in Arabic text which contains different Arabic dialects like Gulf, Egyptian, Levantine and Maghrebi besides MSA (Modern Standard Arabic). The size of the dataset was 10039 entries collected from Twitter and the web. The text length varied from 3 words minimum up to 446 words maximum. Additionally, the dataset contains 201,095 tokens and 36814 word types. The word type is defined here by counting the number of vocabularies (distinct words) in the corpus.

3.4 Dataset annotation

The annotation guidelines were explained to the annotators through online and face to face meetings. The given guidelines were as follows:

- The labels are defined to the annotators as: Humor - The author has written a comic or amusing text. Non-Humor - The author has not written comic or amusing text.
- Perspective: The text should be considered humorous or not from the annotator's perspective.
- Ambiguity: If the text is not understandable or you cannot get the joke, check non-humor, also if it does not have any keyword related to humorous aspects.
- Incomplete text is caused by retrieving tweets without their corresponding memes, so if the text conveys any funny, amusement or humorous meaning check humor.
- The context of the text could be understood from certain keywords in the text that are related to certain groups or subjects that are known to be subjected to humor examples such as (واحد محشش، قروي، صعيدي، فيه واحد بخيل). Table 3 shows examples of each label.

We tried as much as possible to help annotators in their decision of labelling the data by providing certain guidelines. However, it is difficult to identify something that has no standard definition and ambiguously stated (Castro et al. 2016). In general text comprised of any tales, imaginative, jokes, and abusive are explained as humorous unless it doesn't convey any delightful, entertaining or amusement as in (Khandelwal et al. 2018).

As mentioned above people have different sense of humor so deciding whether the text is humorous or not will be best if it is from the annotator perspective as in (Castro et al. 2016). Since the text collected in MSA and DA (Dialectal Arabic) are different and some of them cannot be understandable by the annotators especially if they were in Magrabi dialect, so we think that the text can be cracked by some known keywords that mostly used in humorous context.

Labels	Examples in Arabic	English Translation
Humor: Something written to be comical or funny	-فيه قروي شاف نافورة جاب لها سباك	- A villager saw a fountain and brought it a plumber.
	-صعيدي راح ل ماكدونالدز وقال لهم عندكم حلقات بصل قالوا ايه قال عطوني الحلقة الأخيرة	- An upper egyptian went to McDonald's and asked them do you have onion rings? ", They said - yes- he said: give me the last episode.
	-في واحد محشش جالس امام المكيف يقوله لا تنفخ لا اكسر راسك	- A stoned sitting in front of the air-conditioner saying: Do not blow or , I will smash your head.
	-فيه واحد بخيل قال الي اولاده اذا نجحو با وريكم سيارة الأيسكريم	-There is a miser told his children that: If you succeed, I will show you the ice cream car.
Non-Humor: Something written not to be humorous or funny	اللي جابو لنا امراض نفسيه عاملين فيها دكاترة نفسيين	Those who brought us Psychiatric illness act like psychiatrists

Table 3: Examples of Humor and non-Humor text in the dataset.

Many tweets contain an image (memes) with a corresponding label that results in losing the full context, here the decision can be made depending on the available text (label or caption) if it conveys any humorous aspect or not, or if it can be understood without the image so it is considered humor, if it is not, it will be non-humor.

Inter-Annotator Agreement (IAA) is a measurement used for the reliability and credibility of the annotators to ensure the quality of the annotation process. It compares the annotators' labels of the text and shows how much annotators agreed upon a particular text or disagreed.

Moreover, it gives insights into how challenging it was to identify each text label and the reasons behind this labeling decision. Also, it indicates whether the annotators fully understand the guidelines given to them or not (Khandelwal et al. 2018).

Since we have three annotators for each text, we choose Fleiss's Kappa (Fleiss, 1971) to measure IAA in our annotated corpus. We calculated Fleiss's Kappa for the 10039 texts and the result of Kappa was 0.73 which is considered substantial according to Kappa statistics (Landis and Koch, 1977) .

Table 4 shows the statistics of the annotated dataset which indicates that the dataset is almost balanced because there is no significant difference between the two classes.

Class	Number of entities	Number of Tokens	Percentage
Humor (1)	4455	86989	44.38%
Non-Humor (0)	5584	114171	55.62%
All	10039	201095	100%

Table 4: Distribution of Humor and non-Humor text in the dataset.

4 Experiments and Results

4.1 Model selection

In this study we selected seven Arabic Pre-Trained Language Models based on their performance on other text classification tasks for Arabic language and also being pre-trained on dialectal Arabic (DA). The models are:

1. **AraBERTv02**[6] is a transformer-based model for Arabic language based on the BERT-Base model.
   AraBERTv02 was trained on the Arabic version of the Books Corpus and the Arabic Wikipedia. The model achieves a state-of-the-art performance on a number of Arabic natural language understanding tasks, including question answering and natural language inference.
2. **Arabertv02-twitter**[7] same as AraBERTv02 and trained on ~60M Arabic tweets (filtered from a collection on 100M).
3. **QARIB**[8]: QCRI Arabic and Dialectal BERT (QARiB) model, was trained on a collection of ~ 420 Million tweets and ~ 180 Million sentences of text.
4. **MarBERT**[9] **:** is a large-scale pre-trained masked language model based on the BERT-Base model and focused on both Dialectal Arabic (DA) and MSA.
5. **MARBERTv2**[10] : same as MarBERT with further pre-training the stronger model, on the same MSA data as ARBERT in addition to AraNews dataset.
6. **CAMeLBERT-DA**[11]: is a BERT model pre-trained on 54GB of dialectal Arabic (DA).
7. **CAMeLBERT-MIX**[12]: is a BERT model pre-trained on 167GB of mix text including Modern Standard Arabic (MSA), dialectal Arabic (DA), and classical Arabic (CA).

The selected models were fine-tuned to perform a binary classification task to detect a given text if it is Humor or Non-Humor.

The dataset was split 85:15 for training and testing respectively using scikit-learn library[13].

4.2 Models' Results

PLM Model	Accuracy	Precision	Recall	F1-score
AraBERTv02	68.72	69.06	69.19	68.71
Arabertv02-twitter	70.91	72.13	71.86	70.89
QARIB	67.72	68.03	68.16	67.71
MarBERT	70.91	72.13	71.86	70.89
MARBERTv2	71.71	72.94	72.67	71.69
CAMeLBERT-DA	**72.11**	**72.75**	**72.79**	**72.11**
CAMeLBERT-MIX	70.31	72.55	71.64	70.20

Table 5: Results of various models trained and tested on our dataset

The selected models were evaluated using different metrics which are accuracy, precision, recall and F1 scores as shown in Table 5. The results of the four metrics show the performance of each model on the testing set. CAMeLBERT-DA

[6] https://huggingface.co/aubmindlab/bert-base-arabertv02

[7] https://huggingface.co/aubmindlab/bert-base-arabertv02-twitter

[8] https://huggingface.co/qarib/bert-base-qarib

[9] https://huggingface.co/UBC-NLP/MARBERT

[10] https://huggingface.co/UBC-NLP/MARBERTv2

[11] https://huggingface.co/CAMeL-Lab/bert-base-arabic-camelbert-da

[12] https://huggingface.co/CAMeL-Lab/bert-base-arabic-camelbert-mix

[13] https://scikit-learn.org

obtained the highest scores in all specified metrics followed by MARBERTv2, this is mainly due to the increase of the vocabulary size and the amount and type of the training data i.e. dialectal Arabic (DA). The performance of Arabertv02-twitter and MarBERT were the same in all metrics. The CAMeLBERT-MIX model came next with relatively small difference, lastly QARIB came with the lowest scores in all metrics.

Comparing our task to other similar works are not feasible as they use different datasets and different languages. Since our task is the only one using this dataset which was collected for this task with the goal of classifying humor in Arabic. Therefore, we developed a baseline model using different measures.

5 Challenges and Issues

As mentioned early the task of humor detection is challenging in general and it is even harder in Arabic language due to the nature of the language. In general, detecting humor needs more knowledge to fully understand it (Khandelwal et al. 2018). The nature of humor also is a challenge by itself as people perceived humors differently, what might be funny to one might not be for another. Lacking a benchmark for humor detection make it difficult to evaluate models' performance. Moreover, one of the main causes of the lag in progress on humor research is the scarcity of public datasets (Hossain et al. 2019).

For Arabic language, (Biniz et al. 2018) stated that analyzing and automating tasks in Arabic is difficult than other languages due to: its morphological richness, its complex syntax, and its difficult semantics. The main challenge of our work was having multiple dialects that are different in syntax that make the detection harder.

6 Conclusion

This study discussed the methods we used to collect and construct humor dataset of different Arabic dialects and MSA. The dataset contained 10039 tweets and was annotated with two labels humor and non-humor. Three annotators have manually annotated the corpus, and Fleiss's Kappa was calculated to ensure the quality of the annotation process. Also, we carried out a set of experiments using a number of PLMs on the dataset to test it. The best model was CAMeLBERT-DA with an accuracy and an F1 score of 72.11%, which indicates potential for improvement and reflects the problematic nature of the humor dataset and the challenge of humor detection in general.

As a future work, we plan to understand and identify which features are essential for humor detection that contribute to enhancing the models prediction and troubleshooting unexpected model outputs.

References

Altin, Lutfiye Seda Mut. 2019. "LaSTUS/TALN at HAHA: Humor Analysis Based on Human Annotation." 6.

Amin, Miriam, and Manuel Burghardt. 2020. "A Survey on Approaches to Computational Humor Generation." Pp. 29–41 in *Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*. Online: International Committee on Computational Linguistics.

Annamoradnejad, Issa, and Gohar Zoghi. 2020. *ColBERT: Using BERT Sentence Embedding for Humor Detection*.

Banikalef, Eddin Abdullah Ahmed. 2014. "Linguistic Analysis of Humor in Jordanian Arabic among Young Jordanians Facebookers." Retrieved December 8, 2021 (https://www.semanticscholar.org/paper/Linguistic-Analysis-of-Humor-in-Jordanian-Arabic-Banikalef/e6aaed4aeb4e627678b62203791d959c792dd625).

Bertero, Dario, and Pascale Fung. 2016. "Multimodal Deep Neural Nets for Detecting Humor in TV Sitcoms." Pp. 383–90 in *2016 IEEE Spoken Language Technology Workshop (SLT)*.

Biniz, Mohamed, Samir Boukil, Fatiha Adnani, Loubna Cherrat, and Abd Moutaouakkil. 2018. "Arabic Text Classification Using Deep Learning Technics." *International Journal of Grid and Distributed Computing* 11:103–14. doi: 10.14257/ijgdc.2018.11.9.09.

Bzour, Assistant Prof Anwar Fayez Al. 2021. "Arabic Humorous Texts: An Attempt to Analyze." *Review of International Geographical Education Online* 11(4):1480-1492.

Castro, Santiago, Matías Cubero, Diego Garat, and Guillermo Moncecchi. 2016. “Is This a Joke? Detecting Humor in Spanish Tweets.” *ArXiv:1703.09527 [Cs]* 10022:139–50. doi: 10.1007/978-3-319-47955-2_12.

Fan, Xiaochao, Hongfei Lin, Liang Yang, Yufeng Diao, Chen Shen, Yonghe Chu, and Yanbo Zou. 2020. “Humor Detection via an Internal and External Neural Network.” *Neurocomputing* 394. doi: 10.1016/j.neucom.2020.02.030.

Fleiss, Joseph L. 1971. “Measuring Nominal Scale Agreement among Many Raters.” *Psychological Bulletin* 76(5):378–82. doi: 10.1037/h0031619.

Hasan, Md Kamrul, Wasifur Rahman, Amir Zadeh, Jianyuan Zhong, Md Iftekhar Tanveer, Louis-Philippe Morency, Mohammed, and Hoque. 2019. “UR-FUNNY: A Multimodal Language Dataset for Understanding Humor.” *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* 2046–56. doi: 10.18653/v1/D19-1211.

Hossain, Nabil, John Krumm, and Michael Gamon. 2019. “‘President Vows to Cut <Taxes> Hair’: Dataset and Analysis of Creative Text Editing for Humorous Headlines.” *ArXiv:1906.00274 [Cs]*.

Kao, Justine T., R. Levy, and Noah D. Goodman. 2016. “A Computational Model of Linguistic Humor in Puns.” *Cogn. Sci.* doi: 10.1111/cogs.12269.

Khandelwal, Ankush, Sahil Swami, Syed S. Akhtar, and Manish Shrivastava. 2018. “Humor Detection in English-Hindi Code-Mixed Social Media Content : Corpus and Baseline System.” *ArXiv:1806.05513 [Cs]*.

Landis, J. Richard, and Gary G. Koch. 1977. “The Measurement of Observer Agreement for Categorical Data.” *Biometrics* 33(1):159. doi: 10.2307/2529310.

Li, Da, Rafal Rzepka, Michal Ptaszynski, and Kenji Araki. 2020. “HEMOS: A Novel Deep Learning-Based Fine-Grained Humor Detecting Method for Sentiment Analysis of Social Media.” *Information Processing & Management* 57:102290. doi: 10.1016/j.ipm.2020.102290.

Meaney, J. A., Steven Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. “SemEval 2021 Task 7: HaHackathon, Detecting and Rating Humor and Offense.” Pp. 105–19 in *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*. Online: Association for Computational Linguistics.

Meyer, John C. 2000. “Humor as a Double-Edged Sword: Four Functions of Humor in Communication.” *Communication Theory* 10(3):310–31. doi: 10.1111/j.1468-2885.2000.tb00194.x.

Nayef, Heba, and Mohamed El-Nashar. 2014. “‘Dissecting the Poisoned Honey’ Sexist Humor in Egypt: A Linguistic Analysis of Sexism in Colloquial Cairene Arabic Jokes.” *Anàlisi* 131. doi: 10.7238/a.v0i50.2324.

Omar, Hidaya Moulay. 2017. “A Linguistic Analysis of Humour in Algerian Arabic : The Case of ‘Sultan Ashur X’ Sitcom.” Thesis, Yarmouk University.

Zhang, Dongyu, Heting Zhang, Xikai Liu, Hongfei Lin, and Feng Xia. 2019. “Telling the Whole Story: A Manually Annotated Chinese Dataset for the Analysis of Humor in Jokes.” Pp. 6402–7 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)*. Hong Kong, China: Association for Computational Linguistics.

Zhang, Renxian, and Naishi Liu. 2014. “Recognizing Humor on Twitter.” Pp. 889–98 in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, *CIKM ’14*. New York, NY, USA: Association for Computing Machinery.

A deep sentiment analysis of Tunisian dialect comments on multi-domain posts in different social media platforms

Emna Fsih, Rahma Boujelbane and **Lamia Hadrich Belguith**
ANLP-RG, MIRACL Lab. University of Sfax, Tunisia
{emnafsih, rahma.boujelbane}@gmail.com
lamia.belguith@fsegs.usf.tn

Abstract

With the emergence of social media, the Tunisian Dialect (TD), as the other Arabic dialects, started having a wide representation in the written form. It has switched from a purely oral language to a written form without normalizing or utilizing any orthographic convention or standard. Therefore, it is necessary to investigate these opinions and analyze them in order to extract useful knowledge. For this we propose in this paper an approach to create a richly annotated corpus. Then, by exploiting this corpus, we compare the effectiveness of machine learning and transfer learning models to build fine grained sentiment analysis models. The BERT model, Fine tuned with TD data, achieved the best result.

1 Introduction

Opinion Mining (OM) or Sentiment Analysis (SA) can be defined as the task of detecting, extracting and classifying opinions on some issues/facts (Vinodhini and Chandrasekaran, 2012). Indeed, the proliferation of social media has allowed collecting data for low-resourced languages. Facebook, Instagram and YouTube have become very popular tools for sharing videos and communication. Nowadays, these social networks are intensively used by different kinds of companies to develop their activities and this is through analysing the opinions of internet users. This analysis is constrained by the availability of adequate tools and resources. The dilemma is accentuated when it comes to dealing with a poorly endowed language.

In this work, we process comments posted on content related to news and companies in Tunisia. These comments are mainly written in Tunisian dialect but they could integrate other languages namely Arabic dialects, standard Arabic or other foreign languages and sometimes applying a code switching between two or more languages, which generates a huge orthographic heterogeneity. Thus a word belonging to a such corpus may be an emoji, a word written in Latin script or a word written in Arabic script. Each form allows designating more than one language. Indeed, a word in Latin script can be a word in French (or also in English). Table 1 illustrates an example of comment forms. In this case, the automatic identification of the words written as TD is not trivial. The lexicon in Arabic script can also be either written a standard Arabic word, dialect word or a TD word. We wish through this work to propose a deep analysis for different types of comments present in different social media platforms. For this we propose a method which consists of building a fine-grained annotated corpus and to train different machine-learning and deep-learning models with the resulted corpus. These resources could be later useful and beneficial for other NLP tasks.

The remainder of this paper is organized as follows. Section 2 discusses some related works. Section 3 describes the proposed approach for TD opinion analysis. Section 4 details the Tunisian Dialect sentiment analysis resources. Section 5 presents sentiment analysis models for TD comments. Finally, section 6 concludes and points to possible directions for future work.

Forms	**Comment**
Latin script	t3jbny //I like you//
Arabic script	ممتازة //Excellent//
Emoji	♥ ✌
Latin script in French	trés bonne //very good//
Latin script in dialect	mo7tarem //respectable//

Table 1: Comment forms.

2 Related work

Studies on sentiment analysis of Arabic dialect in social networks have rapidly grown in recent years. Several researchers have attacked the treatment of this field on several aspects and levels.(Alahmary et al., 2019) for example, in order to develop sentiment and emotion annotation Twitter corpus for Saudi dialect, have created SDCT a corpus in Saudi dialect by annotating 32,063 tweets into two classes containing 17707 positive tweets and 14356 negative tweets. Using these resources, the authors have compared the effectiveness of the LSTM, Bi-LSTM models and the SVM model. The experimental results have showed that, in their context, the use of Bi-LSTM (F_score 94%) is more efficient than the use of LSTM and SVM.

Also, (Mohammed and Kora, 2019) have applied three Deep Learning models (DL): CNN, LSTM and RCNN in a network that combines some of the characteristics of both CNN and LSTM neural networks in the sense that CNN is used as a strong feature extractor and LSTM layer applies the recurrent neural network architecture on those extract features on 40k Arabic tweets for Arabic sentiment analysis. The best accuracy achieved was 88% by LSTM model. In addition (Al-Smadi et al., 2019) proposed a method for Arabic sentiment analysis. Authors have used LSTM implementation. Experiments have been done on Arabic hotels' reviews dataset as (19,226) for training and (4802) for testing and recorded an F1-Score of 82.6%. Similarly, (Dahou et al., 2019) used CNN algorithm to perform Arabic sentiment analysis, experimental results were evaluated on five different Arabic sentiment datasets. The best accuracy achieved was 93.28%. Recently, research has gone beyond deep learning approaches and explored the advances offered by transfer learning using transformer architectures based on the encoder-decoder pattern. (Moudjari et al., 2020) evaluates deep learning models (LSTM, CNN and BERT), to classify if an Algerian tweets (9000 tweets) as either positive, negative or neutral. The best results in term of accuracy were obtained with the CNN and LSTM models with 76% and 75%, respectively. On the other hand, BERT gave the worst results in term of accuracy with 68%.

In (Abdul-Mageed et al., 2020) implementing ALBERT and MARBERT models: i) ARBERT is a large scale pre-training masked language model focused on Modern Standard Arabic (MSA). ii) MARBERT is a large scale pre-training masked language model focused on both Dialectal Arabic (DA) and MSA. Both models implemented for multiple text classification tasks: (1) sentiment analysis (SA), (2) social meaning (SM), (3) topic classification (TC), (4) dialect identification (DI), etc in Arabic. The sentiment analysis model achieved the F-score of 71.50% when applied with MARBERT model. Similarly, (Abuzayed and Al-Khalifa, 2021) proposed sentiment detection for Arabic dialect language by augmenting data proposed by the shared task in (Abu Farha et al., 2021) to analyse the sentiment of tweets. By using the MARABERT model, they obtained an F1-score of 86%. (Abdel-Salam, 2021) have also fine-tuned MARBERT for sentiment classification tweets in Arabic dialect: MSA, Egyptian, Maghrebi dialect, etc and MARBERT model achieved an accuracy of 69.57%.

Sentiment analysis for Tunisian dialect: To develop sentiment analysis model for Tunisian Dialect (TD), many efforts have been made to develop resources such as annotated corpus, lexicon and models. (Mdhaffar et al., 2017) have created the Tunisian Sentiment Analysis Corpus (TSAC) from Facebook official pages of Tunisian radios and TV channels. TSAC contains 17k comments written only with Arabic letters and extracted from Facebook. This corpus has been annotated by a native speaker with 8215 positive and 8845 negative comments. An MLP classifier was then applied to build a sentiment analysis model that achieved an F1-score of about 78%. The TunBERT (Messaoudi et al., 2022) was trained on a TSAC dataset (Mdhaffar et al., 2017) including 7452 comments on Tunisian dialect in Social media and Tunisian Election Corpus (TEC) (Sayadi et al., 2016) 3042 tweets obtained from twitter about Tunisian elections in 2014. It is composed of MSA and Tunisian content. It achieved great results on the TSAC corpus with an accuracy of 96.98% compared to 81.2% on the TEC corpus.

3 Proposed approach

Given the orthographic heterogeneity of the Tunisian dialect in social networks we propose in this work, a process to build deeper Sentiment Analysis (SA) models. For this we exploited, at first, the resources publicly available in the state of the art and we endowed them with more annotations. Using these resources we tried to learn different sentiment analysis models with the aim of

Corpus	TSAC	TSAC+	TSAC+	
		Facebook	YouTube	Instagram
Number of comments	17060	16000	49000	7000
Number of words	113196	45536	126288	11962
Number of vocabulary items	42123	40809	97460	9940

Table 2: Statistics about the corpus.

	LEX1	LEX2	LEX3	LEX4
Simple words	227	312	3213	4041
Phrasal	331	643	7385	7673
Foreign words	11	12	19	45
Foreign phrasal	7	26	34	66
Emoticons	0	0	0	61
Positive opinion indicator	283	419	3638	4581
Negative opinion indicator	293	589	6960	7775
Total	576	1035	10651	12356

Table 3: Statistics of TD sentiment analysis lexicon.

providing comments with more analysis tags. In addition, to treat Arabic dialects comments that may be presented to comment publications intended for Tunisian people, we tested and compared the performance of the TD model and the model learned on a corpus augmented with such content. We detail in what follows the different stages of this process.

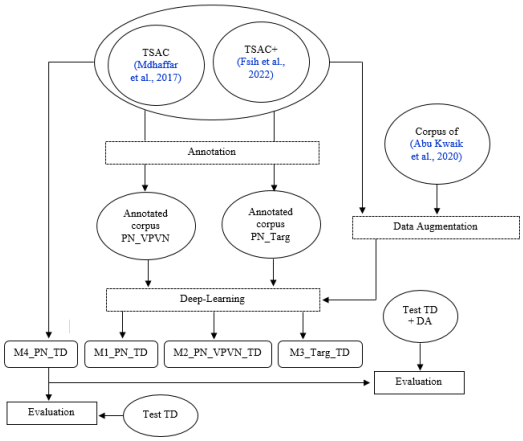


Figure 1: The proposed process to develop fine grained SA models

4 Tunisian Dialect Sentiment Analysis Resources

TSAC: We exploited the TSAC (Tunisian Sentiment Analysis Corpus) proposed by (Mdhaffar et al., 2017). This corpus contains 17k comments written with Arabic letters. It has been annotated into 8212 positive comments and 8854 negative comments of opinions expressed on Facebook and taken from Tunisian TV and radio pages during the period from January 2015 to June 2016.

TSAC+: It is a corpus developed by (Fsih et al., 2022). It contains 65K comments scraped from Tunisian TV channels during a period spanning over January 2016 through March 2019. In order to broaden the spectrum of sentiment analysis, we gathered by 7k comments containing opinions on content presented by influence's from different fields on Instagram. TSAC+ has been annotated into positive (42k) and negative (30k) comments. It has been annotated using TD lexicon. More statistics on the corpus are shown in table 2.

Corpus of Arabic dialect: Arabic Tweets Sentiment Analysis Dataset (ATSAD) presented in (Abu Kwaik et al., 2020) collected from Twitter consists of 42k tweets which are classified as positive and negative, which is available at `https://github.com/motazsaad/arabic-sentiment-analysis`. Table 4 shows the statistics of the corpus.

TD Sentiment analysis lexicon: These resources have also been developped by (Fsih et al., 2022). They contain 12356 entries classified into two main categories: simple words and phrases associated with their inflected forms (**LEX2**) and also with the corresponding ambiguous forms (**LEX3**). The following example shows the different forms corre-

Number of tweets	42693
Number of words	439518
Number of vocabulary items	64629
Number of emoji's	455
Number of positive tweets	18106
Number of negative tweets	24588

Table 4: The statistics of the Arabic dialect tweets.

sponding to the lexical entry "ناجح" //successful// (See Figure 2). Based on this process, we enriched the lexicon with new entries taken from TSAC+ (**LEX4**). Table 3 shows details about the TD sentiment analysis lexicon.

Figure 2: Example of lexicon forms

4.1 Fine grained sentiment analysis annotation

Polarity Opinion annotation: The idea of this paper is to provide a deep analysis to the TD comments, for this and using the TD sentiment analysis lexicon we have annotated the corpus of TSAC+ using the following logic : we calculate for each comment the number of occurrences of positive and negative words. If the number of occurrences of positive words is greater than the negative words, the comment will be annotated as very positive and vice versa as very negative. For example, the comment "magnifique bravo stevy aymen" //wonderful bravo stevy aymen// contains two positive opinion indicators ("magnifique" //wonderful//, bravo) and zero negative indicators, the number of occurrences of positive words is greater than negative, the comment is then annotated as very positive. The table below illustrates the opinion annotation statistics with degree of polarity.

	TSAC+
Number of positive comments	20170
Number of negative comments	18115
Number of comments very positive	20794
Number of comments very negative	13630

Table 5: Statistics of the annotation with polarity.

Target annotation: We also sought to annotate the target of the opinion, for this, we extracted,at first, different target indicators like " حلقة "//Episode//, " برنامج" //Program//, "قناة" //Channel// from TSAC+ corpus. Five target categories "episode, program, person, subject and channel" were distinguished. Each category has its corresponding indicators. For instance, the category person has two types of indicators: 305 of them are named entities and 901 are adjectives used to designate a person such as "منشط" //A television presenter//, "ضيف" //A guest//, "مذيع" //Broadcaster//. Comments having no indicators were classified as neutral. Table 6: reveals the number of comments in each target category.

Categories	**Number of comments**
Episode	755
Program	1349
Subject	183
Channel	614
Person	36570
Neutral	33250

Table 6: Statistics of target comments.

Comments	**Polarity**	**Target**
ya m3alem ya sami bravo //Great sami bravo//	Positive	Person
حلقة عالمية //Great episode//	Positive	Episode
ملا تفاهة //What is this nonsense//	Negative	Natural
برنامج فاشل //Failed program//	Negative	Program

Table 7: Examples of TD comments.

5 Sentiment Analysis Models for TD Comments

5.1 Standard sentiment analysis model (M1_PN_TD):

Most of the works cited in the state of the art have focused on the construction of the usual sentiment analysis models, i.e they only detect Positive (P) and Negative (N) polarities. For this, we first used 59K comments of the corpus cited in the previous section to build a model able to detect Positive and Negative TD comment (M1_PN_TD). For this, we

M1_PN_TD			
Polarity	**Precision**	**Recall**	**F-measure**
Positive	72%	91%	81%
Negative	81%	53%	64%
Accuracy		75%	
M4_PN_AD			
Polarity	**Precision**	**Recall**	**F-measure**
Positive	83%	92%	87%
Negative	87%	75%	81%
Accuracy		85%	

Table 8: Evaluation of BERT model for Arabic dialect tweets classification.

M1_PN_TD Machine-learning model				
Classifiers	**Polarity**	**Precision**	**Recall**	**F-measure**
SMO	Positive	84%	96%	89%
	Negative	88%	61%	72%
NB	Positive	82%	98%	89%
	Negative	93%	54%	69%
DT	Positive	81%	94%	87%
	Negative	81%	55%	66%
Accuracy		SMO=84.66%	NB=83.85%	DT=81.13%
M1_PN_TD Deep-learning model				
Polarity		**Precision**	**Recall**	**F-measure**
Positive		84%	96%	90%
Negative		89%	63%	74%
Accuracy			86%	

Table 9: Evaluation of Machine-learning and Deep-learning models for Positive-Negative classification.

examined at first the effectiveness of three different text mining algorithms including SMO, Naive Bayes (NB) and Decision Tree (DT) model. Moreover, given the efficiency of transformer architecture on different NLP tasks, we fine tuned a BERT (Bidirectional Encoder Representations from Transformers) model learned on dialectal data (Fsih et al., 2022). The different models were based on a test corpus containing 13000 sentences. Results in table 9 show that the BERT model gives the best result with an accuracy of 86%.

5.2 Fine grained sentiment analysis models

Deep sentiment analysis model (M2_PN_VPVN_TD): We used the corpus annotated with 4 sentiments tags: Very Positive (VP), Very Negative (VN), positive and negative tags to study a sentiment analysis model. The same algorithms as before have been adopted. The best result was obtained using the BERT algorithm with an accuracy of 77% (Table 10).

Target of opinion detection model (M3_Targ_TD): On the corpus annotated with Target (Targ) tags, we also learned, using the same algorithms as the other experiments, a machine and deep-learning models for the detection of the target of the opinion. The resulted models have been projected on the same test corpus used in the experiments presented in this paper. The obtained results are given in table 11. The BERT model achieves the higher results with an accuracy of 96%.

multi-Arabic Dialect Deep sentiment analysis model (M4_PN_TD_AD): We wish through this experiment, studying the effect of the models designed for the TD on a mixed content including TD comments and dialectal comments. For this we introduced at first to the test corpus 21k comments (8k comments belonging to the different Arabic dialects and 13k comments in TD) on which we projected the model (M1_PN_TD). Then we introduced to the training data, dialectal comments con-

M2_PN_VPVN_TD Machine-learning model				
Classifiers	**Polarity**	**Precision**	**Recall**	**F-measure**
SMO	Positive	58%	79%	67%
	Negative	64%	35%	46%
	Very-Positive	73%	61%	66%
	Very-Negative	54%	66%	60%
NB	Positive	59%	36%	45%
	Negative	61%	51%	56%
	Very-Positive	50%	83%	63%
	Very-Negative	70%	42%	52%
DT	Positive	54%	80%	64%
	Negative	53%	35%	42%
	Very-Positive	72%	51%	60%
	Very-Negative	54%	48%	51%
Accuracy		SMO=62.56%	NB=55.33%	DT=57.85%
M2_PN_VPVN_TD Deep-learning model				
Polarity		**Precision**	**Recall**	**F-measure**
Positive		74%	86%	80%
Negative		71%	45%	55%
Very-positive		86%	89%	87%
Very-negative		67%	71%	69%
Accuracy			77%	

Table 10: Evaluation of Machine-learning and Deep-learning models for VeryPositive-VeryNegative classification.

taining 14k positive and 19k negative comments and then we compared the results. Table 8 shows the obtained results. We noticed that the augmentation performed with the dialect data improves the results. The model trained on this corpus achieves an accuracy of 85%.

6 Conclusion

In this paper, we presented an approach to build resources to analyse opinions written in Tunisian dialect. Indeed, we built a corpus containing 72k commentaries annotated in polarity and opinion target. Then, we developed four automatic opinion analysis models. The first allows to predict if the opinion is positive or negative. The second predicts whether the opinion is positive, very positive, negative or very negative. The third makes it possible to predict the target of the opinion. We expanded the spectrum of lexical coverage at the level of the fourth model. All these resources will be open for access once the paper is published. In future work we aim to test the effectiveness of other transformer models and to further extend the lexical coverage of the model to cover more ambiguities such as code switching or other languages.

References

Reem Abdel-Salam. 2021. WANLP 2021 shared-task: Towards irony and sentiment detection in Arabic tweets using multi-headed-LSTM-CNN-GRU and MaRBERT. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 306–311, Kyiv, Ukraine (Virtual). Association for Computational Linguistics.

Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. 2020. Arbert & marbert: deep bidirectional transformers for arabic. *arXiv preprint arXiv:2101.01785*.

Ibrahim Abu Farha, Wajdi Zaghouani, and Walid Magdy. 2021. Overview of the WANLP 2021 shared task on sarcasm and sentiment detection in Arabic. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 296–305, Kyiv, Ukraine (Virtual). Association for Computational Linguistics.

Kathrein Abu Kwaik, Stergios Chatzikyriakidis, Simon Dobnik, Motaz Saad, and Richard Johansson. 2020. An Arabic tweets sentiment analysis dataset (ATSAD) using distant supervision and self training. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 1–8, Marseille, France. European Language Resource Association.

Abeer Abuzayed and Hend Al-Khalifa. 2021. Sarcasm and sentiment detection in arabic tweets using bert-

M3_Targ_TD Machine-learning model				
Classifiers	**Target**	**Precisison**	**Recall**	**F-measure**
SMO	Person	87%	91%	89%
	Program	98%	83%	90%
	Episode	99%	87%	93%
	Subject	78%	47%	58%
	Channel	94%	90%	92%
	Neutral	93%	92%	92%
NB	Person	53%	98%	68%
	Program	97%	91%	94%
	Episode	95%	81%	88%
	Subject	98%	47%	38%
	Channel	94%	89%	92%
	Neutral	91%	48%	63%
DT	Person	91%	83%	87%
	Program	97%	85%	91%
	Episode	98%	84%	91%
	Subject	80%	40%	53%
	Channel	90%	89%	89%
	Neutral	89%	95%	92%
Accuracy		SMO=90.39%	NB=64.40%	DT=90%
M3_Targ_TD Deep-learning model				
Target		**Precision**	**Recall**	**F-measure**
Person		94%	96%	95%
Program		98%	94%	96%
Episode		97%	92%	95%
Subject		80%	80%	80%
Channel		99%	97%	98%
Neutral		97%	96%	97%
Accuracy			96%	

Table 11: Evaluation of Machine-learning and Deep-learning models for target classification.

based models and data augmentation. In *Proceedings of the sixth Arabic natural language processing workshop*, pages 312–317.

Mohammad Al-Smadi, Bashar Talafha, Mahmoud Al-Ayyoub, and Yaser Jararweh. 2019. Using long short-term memory deep neural networks for aspect-based sentiment analysis of arabic reviews. *International Journal of Machine Learning and Cybernetics*, 10(8):2163–2175.

Rahma M Alahmary, Hmood Z Al-Dossari, and Ahmed Z Emam. 2019. Sentiment analysis of saudi dialect using deep learning techniques. In *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–6. IEEE.

Abdelghani Dahou, Mohamed Abd Elaziz, Junwei Zhou, and Shengwu Xiong. 2019. Arabic sentiment classification using convolutional neural network and differential evolution algorithm. *Computational intelligence and neuroscience*, 2019.

Emna Fsih, Rahma Boujelbane, and Lamia Hadrich Belguith. 2022. Resources building for sentiment analysis on content disseminated by tunisian media in social networks. *Manuscript submitted for publication.*

Salima Mdhaffar, Fethi Bougares, Yannick Esteve, and Lamia Hadrich-Belguith. 2017. Sentiment analysis of tunisian dialects: Linguistic ressources and experiments. In *Third Arabic Natural Language Processing Workshop (WANLP)*, pages 55–61.

Abir Messaoudi, Ahmed Cheikhrouhou, Hatem Haddad, Nourchene Ferchichi, Moez BenHajhmida, Abir Korched, Malek Naski, Faten Ghriss, and Amine Kerkeni. 2022. *TunBERT: Pretrained Contextualized Text Representation for Tunisian Dialect*, pages 278–290. arXiv.

Ammar Mohammed and Rania Kora. 2019. Deep learning approaches for arabic sentiment analysis. *Social Network Analysis and Mining*, 9(1):1–12.

Leila Moudjari, Karima Akli-Astouati, and Farah Benamara. 2020. An Algerian corpus and an annotation platform for opinion and emotion analysis. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1202–1210, Marseille, France. European Language Resources Association.

Karim Sayadi, Marcus Liwicki, Rolf Ingold, and Marc Bui. 2016. Tunisian dialect and modern standard arabic dataset for sentiment analysis: Tunisian election context. In *Second International Conference on Arabic Computational Linguistics, ACLING*, pages 35–53.

G Vinodhini and RM Chandrasekaran. 2012. Sentiment analysis and opinion mining: a survey. *International Journal*, 2(6):282–292.

TuniSER: Toward a Tunisian Speech Emotion Recognition System

Abir Messaoudi
iCompass
abir@icompass.digital

Hatem Haddad
iCompass
hatem@icompass.digital

Moez Benhaj Hmida
iCompass
moez@icompass.digital

Mohamed Graiet
ISIMM
mohamed.graiet1@gmail.com

Abstract

Speech Emotion Recognition (SER) has become an important component for Human-Computer interaction. It is generally used in job interviews, caller-agent calls and streaming videos, etc. In the speech emotion recognition literature, many languages have tackled this topic to extract emotions from signals. The purpose of this work is to build a Speech Emotion Recognition model that predicts the emotional state of Tunisian speakers. We explore different pre-trained acoustic models, we detail the process of building the first Tunisian Speech Emotion Recognition dataset (TuniSER) and we describe the training and testing phases. Our experiments' results show that fine-tuning the pretrained multilingual wav2vec 2.0 model on the Automatic Speech Recognition downstream task then building a classifier on top of fit outperformed all the tested models achieving an Accuracy of 60.6%.

1 Introduction

In a conversation, non-verbal communication contains important information such as the speaker's intentions or emotions. This information needs to be processed and recognised. Indeed, speech systems should be able to understand this non-linguistic information. In the recent years, the interest in Speech Emotion Recognition (SER) has increased due to the role that this task plays in improving both the naturalness and efficiency of human-machine interactions. Voice assistants and conversational interfaces have become omnipresent through technology devices such as smartphones and smart home interaction systems. Once these systems capture the emotional content of speech aside from semantics, their capability will increase.

SER is a non-trivial task on account of many reasons such as the ambiguity of defining emotions itself and the non-obvious ability of detecting the natural from the acted emotions. Also, it requires large annotated emotional datasets. Yet, creating such data is cost prohibitive because of the large human efforts involved.

Moreover, SER becomes extremely a complicated task with an under-resourced language like the Tunisian dialect (Fourati et al., 2020) because of the lack of resources and the non-existence of the emotional Tunisian dataset. There are various existing researches in the field of Arabic Speech Emotion Recognition. But, they are basically restricted to MSA (Hifny and Ali, 2019).

The primary research questions we wish to investigate in the paper are:

- Question 1: How to build a labeled speech emotional dataset when it comes to under-resourced dialect?
- Question 2: Can the SER model identify the emotional state of a person regardless of the language or dialect used?
- Question 3: Can we use large multilingual pre-trained acoustic models to classify Tunisian dialectal emotional states?
- Question 4: Which mathematical model for creating reliable recognizers in the case of Tunisian dialect?

The paper structure is described as follows. In Sections 2 we introduce the related work on English and Arabic Sentiment Emotion Recognition. In Section 3 we describe the different steps to build a SER Tunisian dataset. In Section 4, we present the proposed methods to build a SER model. In Section 5 we detail the different experiments and present the outcomes, and finally, the paper is concluded in Section 6.

2 Relation to prior work

For Speech Emotion Recognition problems, different methods have been used such as SVM (Seehapoch and Wongthanavasu, 2013), HMM

(Schuller et al., 2003), decision-trees (Lee et al., 2011) etc. These methods explored different features that detect the emotion held in speech including pitch, shimmer, jitter and MFCCs (Mel-Frequency Cepstral Coefficients) (Ghosh et al., 2016) (Liu et al., 2018). However, one of the major drawbacks of these techniques is that they require a previous knowledge of all the mandatory features that had a direct impact on the emotion recognition task like energy and fundamental frequency (F0). A work done (Sahu, 2019) has revealed that traditional machine learning methods still can reach a high performance as the latest Deep learning models (such as LSTM) in this field.

Deep neural networks represent the latest models in Speech Emotion Recognition. They were used to automatically extract high-level features from audio and have successfully achieved high performances (Han et al., 2014). Since then, different neural network architectures have been used for Speech Emotion Recognition. An innovative study (Zheng et al., 2015) was done by applying CNN for speaker independent emotion recognition systems. They came up with the conclusion that deep learning methods outperform machine learning methods for SER.

Other successful methods were deployed such as RNN and bidirectional long-short term memory (BLSTM) (Lee and Tashev, 2015). Another attempt was to combine a CNN model with a RNN model (Trigeorgis et al., 2016) and it has shown a success as well and an efficient speech emotion recognition. Artificial Neural Networks (Shaw et al., 2016), Deep Convolutional Neural Networks (Zhang et al., 2017) and other deep learning approaches (Abbaschian et al., 2021) were used to bring out the best result for the SER task.

Recently, wav2vec 2.0 (Baevski et al., 2020) has been used in emotion classification task (Pepino et al., 2021) and results in state-of-the-art results for both IEMOCAP and RAVDESS datasets with a recall of 0.67 and 0.84 respectively.

When it comes to Arabic Speech Emotion Recognition (ASER), there are few available datasets for Arabic language. The Basic Arabic Vocal Emotions Dataset (BAVED) (Aouf, 2019) contains Arabic words spelled in three levels of emotions recorded in an audio format, low emotion (tired or exhausted), neutral emotion, and high emotion positive or negative emotions (happiness, joy, sadness, anger). The dataset contains 1935 recordings that are recorded by 61 speakers (45 males and 16 females).

(Mohamed and Aly, 2021) introduced a recognition model for Arabic speech dialogues based on deep learning . The developed model employs the state of the art audio representations including wav2vec2.0 and HuBERT (Hsu et al., 2021). They reached an accuracy of 89% and 87% respectively for wav2vec 2.0 and HuBERT.

(Meddeb et al., 2016) present the main steps to extract and recognize basic emotions (Neutral, Happiness, Sadness, Anger and Fear) in the Arabic speech. They created an Emotional speech database called REGIM_TES (Meddeb et al., 2014) containing 720 speech samples. The length of speech samples is up to 5 Seconds. The selected features in the study are: Pitch of voice, Energy, MFCCs, Formant, LPC and the spectrogram. Results showed that pooling together features extracted at different sites improved classification performances.

As far as we know, there is no Speech Emotion Recognition dataset for the Tunisian Dialect. In the next section, we present our methodology to build a SER dataset for an under-represented language.

3 Tunisian Speech Emotion Recognition Dataset

Speech datasets used for building Speech Emotion Recognition systems are divided into three types namely:

- **Simulated**: Simulated databases are created by reading the same text by different trained speakers with different emotions. The numbers of distinct emotions are important, as they have synthesized emotions, they are disposed to have over-fitted models around emotions a little bit different than what is happening in real life and day-to-day conversations. Those databases make comparing results very easy due to the standardized collections of emotions. Berlin Database of Emotional Speech[1] and the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) (Livingstone and Russo, 2018)[2], are some simulated datasets used for SER (Abbaschian et al., 2021).

- **Induced**: Semi-natural databases are very

[1] http://emodb.bilderbar.info/start.html
[2] https://smartlaboratory.org/ravdess/

similar to the natural utterances of speech, even if they are made based on scenarios with a contextual setting. The emotions are artificial because speakers know that they are recorded. Unfortunately, those data sets have a limited number of emotions due to the limited cases of the scenarios compared to other types of data sets (Abbaschian et al., 2021)(Zheng et al., 2015).

- **Natural**: The natural datasets are made of fully natural emotions which eliminate the problem of being artificially made. They are very effective because they perfectly represent our daily life due to the contentiousness of emotions and the existence of the background noise and concurrent emotions and the dynamic variation of the speech. However, those characteristics make the detection and the modeling of the emotions more complicated. The number of emotions is limited due to the limited sources (Abbaschian et al., 2021). VAM[3] is one of the most famous natural databases used for SER tasks.

Universal emotions are defined as six categories: happiness, sadness, disgust, fear, surprise, and anger (Collet et al., 1997). Nevertheless, in this study, we focus only on four categories: Happy, Sad and Angry in addition to Neutral because it is largely used in the state of the art (Han et al., 2014) (Zheng et al., 2015).

3.1 Tunisian dialect

Tunisian Arabic (Tunisian), is the set of dialects of Maghrebi Arabic spoken in Tunisia known locally as Derja (Sayahi, 2014). It is used in the daily life and turn out to be the language of online communication since the 1990 like the social media, SMS, and emails etc. Considering Tunisia as a multilingual country, code-switch and mixing Tunisian with other languages as French, English and Modern Standard Arabic in daily speech is a common thing for the Tunisian people (Daoud, 2007). Tunisian dialect contains many varieties differing from a region to another.

Emotions does not feel the Same across different Cultures, they can differ across cultures through our use of language to understand and express our emotions. Tunisian emotional states are quite different to the other non Tunisian speakers, being angry as a Tunisian is not the same as a German or as an Algerian.

3.2 TuniSER Dataset

In order to build the first Tunisian Speech Emotion Recognition dataset (TuniSER), we divide emotions into two categories:

- Primary emotions: These emotions occur the most and they are the most used for the emotion recognition task. These emotions are: Happy, Sad, Angry, Neutral.
- Secondary emotions: Due to the rarity of these emotions and for long term purposes, we chose to keep them for annotation, the emotions are: Fear, Disgust and Surprised.

The first step is to choose the data sources. We focus on Tunisian series and TV programs publicly available online. We are seeking suitable, useful audios that are rich of emotions. Due to the non effectiveness of natural databases on emotion speech recognition, our dataset is then semi-natural (Induced). We select different audios for actors playing their roles in a sequence and manually extract the ones that contain explicit emotional states. We focus on the quality and diversity of the sources. Audios have to be quite clear and contain both male and females, different ages, situations and contexts.

Once audio sequences containing emotions are collected, we convert them to the required format. To take out the best emotional scenes from the audio, we choose to apply segmentation which separates the input audio sequence into small utterances of a range from 0.41 to 15.31 seconds with an average around 1 seconds (figure 1), with a clear expression (the whole word or sentence) that contains a significant emotional state without ignoring parts of the words and without overlap between the speakers.

We do a manual validation step before the annotation process. We only keep audios based on the following criteria:

- Each utterance should contain only one subject talking and contain only one scene.
- Sound should be clear without noise so that you can hear the speaker clearly talking.
- utterances should not contain music, silence or be with a bad quality.

[3] $https://sail.usc.edu/VAM/vam_release.htm$

The annotation process is divided on three Tunisian native speakers. Two female annotators are at a higher education level (Master/PhD), aged 23, and one female, aged 27, working as research Engineer. Labels are divided into the two emotions categories (Primary and Secondary emotions), according this guideline:

- Happiness: an upbeat, pleasant way of speaking/laughing.
- Sadness: Crying/Dampened mood/lack of energy and enthusiasm.
- Anger: such as speaking harshly or yelling.
- Neutral: lack of emotional state/ nothing in particular.

If an annotator finds an utterance that have another emotional state, it can be annotated as one of these secondary emotions: Fear (Such as rapid breathing and trembling voice.), Disgust (Voice expression that shows disgust.) and Surprise (Such as yelling, screaming or gasping).

We also mark the gender of the speaker in a segment (Male or Female) for long term purposes.

In Table 1, we present the number of clips per label.

Emotion	Female	Male	Total
Happy	153	172	325
Angry	230	387	617
Sad	247	127	374
Neutral	426	896	1322
Fear	27	7	34
Surprised	46	42	88
Disgust	2	2	4
Total	1136	1635	2771

Table 1: Statistics of the Tunisian SER dataset.

The obtained dataset is composed of 2771 utterances, with 1136 female utterances and 1635 male utterances. Our data carries semi-natural emotions that contain contextual and situational information. Every utterance holds a single emotion.

Using an unbalanced dataset will generate an over-classification problem for the larger classes. To solve this, we manually balance the data by extracting the same number of samples for each class. Data preparation was done to facilitate the use of our dataset to train the models. Finally, we obtain a final balanced dataset with 1300 samples

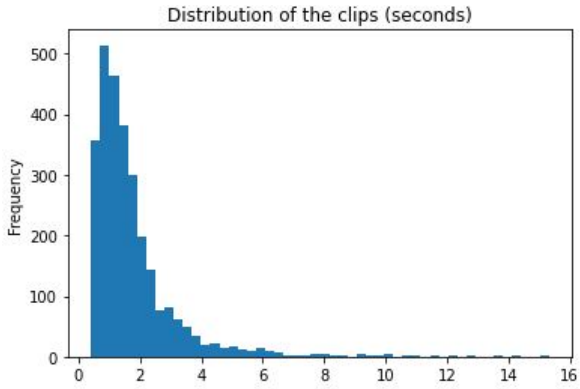


Figure 1: Distribution of the clips length

distributed equally between the four main classes (happy, angry, sad and neutral).

4 Training models

In this section, we will present the architectures of the models used to build our Tunisian SER. In fact, we explored different Machine Learning and Deep learning models for both features extraction and classification steps.

First, we wanted to investigate the influence of languages in SER. We trained using an SVM architecture a SER model on the RAVDESS dataset (Collet et al., 1997) which is an English SER dataset. It includes 1440 audio files of 24 speakers which includes half of male and half female actors and each of them has 60 recordings. The speech is in North-American accent. Then, we used the model to test our in-house Tunisian dialectal dataset. Due to the unsatisfactory results, we deduce that the interpretation of the sense of a word and the emotional states change from a language to another and this is due to multiple factors, such as culture.

Second, we experiment the Long short term memory model (LSTM) with different techniques of feature extraction (Chroma, Mel spectrogram, and Mel-frequency ceptral coefficients). We wanted to investigate which feature works best for the Tunisian SER. We noticed that even if the combination of the 3 features give good results, using only MFCC gives us the best results.

Finally, we introduced two large pre-trained models (The VGGish and wav2vec 2.0 models) as feature extractors followed by classifiers to predict emotions.

4.1 VGGish Model

The google VGGish is a variant of the VGG model (Sahoo et al., 2019) and they have a very similar

architecture. The VGGish model was pre-trained on the AudioSet (Gemmeke et al., 2017) database, which is a collection of more than 2 Million human labeled audio clips collected from Youtube videos with 10 second length each. This database contains over 600 sound classes: Music, Speech, Vehicle, etc. Before feeding the audio clip to the VGGish model, the following steps are applied:

- All audios are resampled to 16 kHz.
- Spectrogram is extracted using magnitudes of the Short-Time Fourier Transform, with three windows: a window size of 25ms, a window hop of 10ms and a periodic Hann window.
- By mapping the spectrogram to 64 mel bins covering the range 125-7500 Hz, a mel spectrogram is computed.
- A stabilized log mel spectrogram is obtained by applying log using the offset to avoid taking a logarithm of zero.
- These features are then framed into non-overlapping examples of 0.96 seconds length, where each example covers 64 mel bands and 96 frames of 10ms each.

Speech segments that are longer than 0.25 carry enough information about the emotional state of the speaker. Therefore, detecting emotion from consecutive segments of the same audio clip will be more efficient (Sahoo et al., 2019). So, we apply overlapping segmentation to catch better correlation between the segments of the same clip and at the same time it is considered as a data augmentation because it increases the number of data points. We extract one-second duration of overlapping segments and fill the last segment with silence to make it one second long. For all segments the overlapping duration is 0.5. This is the first component of our model representing the feature extractor, in which it takes a 96×64 dimensional mel spectrogram as input. VGGish network architecture is constructed of four blocks, two convolution and max-pooling layers followed by 2 fully connected (FC) layers of 4096 units each and finally a FC layer of 128 units that gives the embedding vector. VGGish gives a high-level 128-D embedding from audio input features. Those embedding could be fed to the downstream classification model as input. The VGGish embedding is semantically meaningful and compact than classic and raw audio features

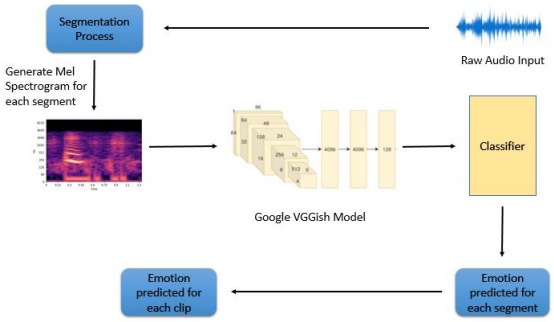


Figure 2: Visual Representation of the VGGish-based model.

so they allow downstream models to be shallower than usual.

For the classification, we applied different approaches based on feeding the 128-dimensional embedding vector to n fully connected layers with N hidden units, with n=1,2 and N =[100,200,400]. Finally, the logit layer is used to predict the emotion for each segment of the same utterance. The full architecture is presented in Figure 2.

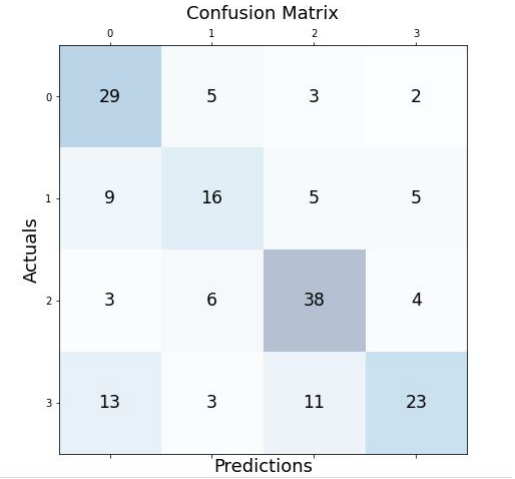


Figure 3: Confusion Matrix with 1 FC layers and number of units=400

4.2 Multilingual wav2vec 2.0

Wav2vec 2.0 (Baevski et al., 2020) is a framework for self-supervised learning of representations from raw audio. It has 2 stages: pre-training and fine-tuning. In pre-training, the speech input is masked in the latent space and a contrastive task with predictions from the transformer and quantized latent speech representations is solved to learn contextualized information. This enables learning powerful representations from speech audio alone. The architecture of wav2vec 2.0 represents three stages,

a local encoder, which contains several convolutional blocks, a contextualized encoder, and a quantization module. To pre-train the acoustic model, we use a multi-layer convolutional feature encoder which takes raw audio as input and outputs latent speech representations. They are then fed to a Transformer to build representations capturing information from the entire sequence. The output of the feature encoder is discretized with a quantization module to represent the targets in the self-supervised objective. The model builds context representations over continuous speech representations and self-attention captures dependencies over the entire sequence of latent representations. We masked a certain proportion of time steps in the latent feature encoder space similar to masked language modeling in BERT (Devlin et al., 2019).

Pre-trained models are then fine-tuned for downstream tasks like Automatic Speech Recognition by adding a classifier with C classes representing the output vocabulary of the respective downstream task on top of the model and training on the labeled data with a Connectionist Temporal Classification (CTC) loss (Graves et al., 2006).

To train our SER model, we used fine-tuned models on both MSA and Tunisian dialect Automatic Speech Recognition downstream task to better determine the context representations for the input audios, since ASR datasets are more available and larger than the SER ones. Finally, we built a neural network as classifier (Table 2) on top of it to predict the class of each audio. The workflow is presented in Figure 4.

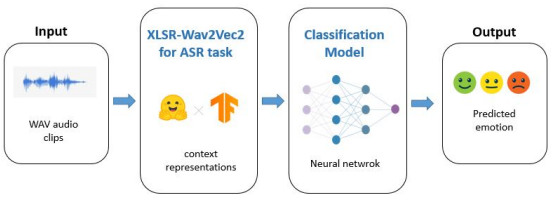


Figure 4: Multilingual Wav2vec 2.0-based model workflow.

5 Experimental Setup and Results

For the VGGish model, we start training our model with the following configuration: 16 as batch size, Adam optimizer with epsilon equal to 10^{-8} and a Learning rate of 10^{-6}. We start our training

Architecture
Dropout
Single layer feed forward network
Tanh
Dropout
Single layer feed forward network

Table 2: Neural Network classifier architecture.

by applying the segmentation process after splitting our data into 70% train set, 15% validation set and 15% test set. We trained the model for 261 epochs, with three different numbers of hidden units N=[100,200,400] and different numbers of FC layers [1,2]. The best result is obtained with 400 numbers of hidden units and 1 Fully connected layer.

For the second approach, we used the multilingual wav2vec 2.0 model. As a first step, we fine-tuned it for the Automatic Speech Recognition downstream task on two languages: Modern Standard Arabic (MSA) and Tunisian dialect. For MSA, the model is trained using the Mozilla Common Voice dataset. For the Tunisian dialect, we used the STAC (Zribi et al., 2015) which is a small Tunisian ASR dataset. Table 3 presents the results of fine-tuning wav2vec on the ASR downstream task on both MSA and Tunisian dialect datasets.

Language	Dataset	WER (%)
MSA	Common voice	52.53
Tunisian Dialect	STAC	62

Table 3: Results of fine-tuning multilingual wav2vec 2.0 on the Automatic Speech Recognition downstream task with MSA and Tunisian Dialect data.

Finally, we built a classification layer on top of the two fine-tuned models to perform the emotions classification. For both models, we adjust the number of epochs and the value of learning rate intuitively to find the best results. We split our balanced data into 80% as training and 20% validation set. 1035 samples for the training and 256 samples for the validation.

6 Discussion

Using the VGGish model as a feature extractor followed by a classifier trained on our constructed Tunisian dialect dataset, we achieved 58,2% as frame accuracy and 60.05% as Average logits clip accuracy, with 1 Fully Connected layer and number

Model	Acc. (%)
LSTM	52.63
VGGish	60.05
multilingual wav2vec MSA	52.10
multilingual wav2vec TD	60.60

Table 4: Comparing different models.

of hidden units = 400. We noticed that changing the number of units gives closer results. The training loss decreased continuously over epochs and the validation loss decreased until the epoch number 151 and started to increase so an early stopping was applied to avoid over-fitting and keep the best result of the model. The confusion matrix, Figure 3, represents the performance of the model for each label of the test dataset. We noticed that adding two Fully Connected layers instead of one layer decreased the results from 58,2% to 54% for the frame accuracy and from 60.05% to 53.1% for the Average logits clip accuracy.

For the second approach, fine-tuning the multilingual wav2vec 2.0 model with MSA and Tunisian dialect gives satisfactory results. In the first try, intuitively running the model with a different number of epochs and learning rate gives 52,1% accuracy with MSA language and 60,6% accuracy with the Tunisian dialect. The training and validation losses for both attempts were decreasing with remarkable fluctuations. These are explained by the fact of using a large neural network with a lot of parameters with small datasets such as our case. This could be solved by increasing the batch size or reducing the parameters of the model.

7 Conclusion and future work

In this paper, we described our methodology to build a Tunisian Speech Emotion Recognition dataset. We detailed the process of using LSTM, and two large pre-trained models: the VGGish and the multilingual wav2vec 2.0 as feature extractors for the Speech Emotion Recognition task. We explained the implementation part for each approach and the different steps followed to train the models. The best result was obtained by fine-tuning the multilingual wav2vec 2.0, which reaches a WER of 60.6%. Our work is an important step for the SER task on the Tunisian dialect, since our satisfactory results could be improved in a future work by augmenting the datasets size and applying enhancement techniques.

References

Babak Joze Abbaschian, Daniel Sierra-Sosa, and Adel Elmaghraby. 2021. Deep learning techniques for speech emotion recognition, from databases to models. *Sensors*, 21(4):1249.

A Aouf. 2019. Basic arabic vocal emotions dataset (baved)–github.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *CoRR*, abs/2006.11477.

Christian Collet, Evelyne Vernet-Maury, Georges Delhomme, and André Dittmar. 1997. Autonomic nervous system response patterns specificity to basic emotions. *Journal of the autonomic nervous system*, 62(1-2):45–57.

Mohamed Daoud. 2007. The language situation in tunisia. In *Language Planning and Policy in Africa, Vol. 2*, pages 256–307. Multilingual Matters.

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.

Chayma Fourati, Abir Messaoudi, and Hatem Haddad. 2020. Tunizi: a tunisian arabizi sentiment analysis dataset. *arXiv preprint arXiv:2004.14303*.

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE.

Sayan Ghosh, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2016. Representation learning for speech emotion recognition. In *Interspeech*, pages 3603–3607.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. volume 2006, pages 369–376.

Kun Han, Dong Yu, and Ivan Tashev. 2014. Speech emotion recognition using deep neural network and extreme learning machine. In *Fifteenth annual conference of the international speech communication association*.

Yasser Hifny and Ahmed Ali. 2019. Efficient arabic emotion recognition using deep neural networks.

In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6710–6714. IEEE.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *CoRR*, abs/2106.07447.

Chi-Chun Lee, Emily Mower, Carlos Busso, Sungbok Lee, and Shrikanth Narayanan. 2011. Emotion recognition using a hierarchical binary decision tree approach. *Speech Communication*, 53(9-10):1162–1171.

Jinkyu Lee and Ivan Tashev. 2015. High-level feature representation using recurrent neural network for speech emotion recognition. In *Sixteenth annual conference of the international speech communication association*.

Zhen-Tao Liu, Min Wu, Wei-Hua Cao, Jun-Wei Mao, Jian-Ping Xu, and Guan-Zheng Tan. 2018. Speech emotion recognition based on feature selection and extreme learning machine decision tree. *Neurocomputing*, 273:271–280.

Steven R Livingstone and Frank A Russo. 2018. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391.

Mohamed Meddeb, K Hichem, and A Alimi. 2016. Automated extraction of features from arabic emotional speech corpus. *International Journal of Computer Information Systems and Industrial Management Applications*, 8:184–194.

Mohamed Meddeb, Hichem Karray, and Adel M Alimi. 2014. Intelligent remote control for tv program based on emotion in arabic speech. *arXiv preprint arXiv:1404.5248*.

Omar Mohamed and Salah A Aly. 2021. Arabic speech emotion recognition employing wav2vec2. 0 and hubert based on baved dataset. *arXiv preprint arXiv:2110.04425*.

Leonardo Pepino, Pablo Riera, and Luciana Ferrer. 2021. Emotion recognition from speech using wav2vec 2.0 embeddings. *arXiv preprint arXiv:2104.03502*.

Sourav Sahoo, Puneet Kumar, Balasubramanian Raman, and Partha Pratim Roy. 2019. A segment level approach to speech emotion recognition using transfer learning. In *Asian Conference on Pattern Recognition*, pages 435–448. Springer.

Gaurav Sahu. 2019. Multimodal speech emotion recognition and ambiguity resolution. *arXiv preprint arXiv:1904.06022*.

L. Sayahi. 2014. *Diglossia and Language Contact: Language Variation and Change in North Africa*. Cambridge Approaches to Language Contact. Cambridge University Press.

Björn Schuller, Gerhard Rigoll, and Manfred Lang. 2003. Hidden markov model-based speech emotion recognition. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 2, pages II–1. Ieee.

Thapanee Seehapoch and Sartra Wongthanavasu. 2013. Speech emotion recognition using support vector machines. In *2013 5th international conference on Knowledge and smart technology (KST)*, pages 86–91. IEEE.

Akash Shaw, Rohan Kumar Vardhan, and Siddharth Saxena. 2016. Emotion recognition and classification in speech using artificial neural networks. *International Journal of Computer Applications*, 145(8):5–9.

George Trigeorgis, Fabien Ringeval, Raymond Brueckner, Erik Marchi, Mihalis A Nicolaou, Björn Schuller, and Stefanos Zafeiriou. 2016. Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5200–5204. IEEE.

Shiqing Zhang, Shiliang Zhang, Tiejun Huang, and Wen Gao. 2017. Speech emotion recognition using deep convolutional neural network and discriminant temporal pyramid matching. *IEEE Transactions on Multimedia*, 20(6):1576–1590.

WQ Zheng, JS Yu, and YX Zou. 2015. An experimental study of speech emotion recognition based on deep convolutional neural networks. In *2015 international conference on affective computing and intelligent interaction (ACII)*, pages 827–831. IEEE.

Inès Zribi, Mariem Ellouze, Lamia Hadrich Belguith, and Philippe Blache. 2015. Spoken tunisian arabic corpus "stac": transcription and annotation. *Research in computing science*, 90:123–135.

Evaluating Large-Language Models for Dimensional Music Emotion Prediction from Social Media Discourse

Patrick J. Donnelly and **Aidan Beery**
Electrical Engineering and Computer Science
Oregon State University - Cascades
Bend, OR, USA
{patrick.donnelly, beerya}@oregonstate.edu

Abstract

The automatic prediction of emotional responses to music is a task of inherent interest to the field of music information retrieval. These efforts are often hindered by the absence of large datasets available for this task. In this work, we investigate the use of sentiment analysis on online social media conversations as an alternate data source to train computational models to predict the emotive responses to a piece of music. Using two datasets annotated with valence and arousal values, we create a corpus of social media commentary for these songs extracted from YouTube, Twitter, and Reddit. We evaluate our approach with transformer models to predict the affective values of the 2402 songs in our dataset. We achieve a moderate Pearson's correlation of 0.62 and 0.72 for valence and arousal, respectively, for discourse from YouTube. These promising results demonstrate that discourse about music may carry semantic information useful to making determinations about the music itself. Such an approach could potentially supplement music information retrieval systems to estimate emotion for pieces of music for which the audio is restricted by copyright or otherwise unavailable.

1 Introduction

The task of music emotion recognition employs computational methods to attempt to predict the emotions elicited by a listener while listening to a piece of music. Estimating the cultural average response of an audience to a song is of interest to the music information retrieval community. A system for automatic music emotion recognition would enable large music libraries to be rated for estimated emotive responses. Online music streaming platforms could then better tune their music recommendation algorithms by filtering by mood or emotion.

Although researchers have investigated many different approaches for music emotion recognition, such efforts have been hindered by the paucity of large datasets suited for this task. These datasets are expensive to create as they require multiple human listeners to manually annotate musical excerpts. Complicating matters, there is no standard definition of this task. Some studies consider four, six, or eight discrete labels used to approximate human emotion. More recent datasets favor the valence-arousal model which treats emotions as a set of continuous values in a multidimensional space (Russell, 1980).

Furthermore, most musical recordings are copyrighted and this usually precludes the release of audio data as part of the dataset. In an attempt to bypass this limitation, the Million Song Dataset[1] released pre-computed acoustic features instead of raw audio. However, this approach limited the ability of researchers to explore certain algorithms or discover innovative features. In the area of music emotion recognition, researchers have struggled with an apparent upper bound in the ability of low-level acoustic features to predict human affective responses to music (Panda et al., 2020). Some researchers have turned to multimodal approaches, such as incorporating natural language analysis of song lyrics to make predictions about the song itself (Laurier et al., 2008).

We hypothesize that social media discussions surrounding a song contain semantic information which can be used to help predict a song's affective qualities. In this work, we present an approach for estimating affective responses to music based solely on commentary from social media. We create datasets of social media discourse for the songs contained in two music emotion datasets. We then compare the efficacy of two popular transformer

[1] http://millionsongdataset.com/

models in the task of predicting affective responses to music trained solely on natural language without the use of signal processing or acoustic features. To our knowledge, this is the first attempt to estimate affective responses to music indirectly based on social media conversations.

2 Background and Related Work

In this section we briefly review some of the approaches for music emotion recognition. We also describe the transformer architectures that we employ in our experiments.

2.1 Acoustic Features

Traditionally, approaches for automatic music emotion recognition have relied on learning information from acoustic features derived from the raw audio of a song. One early approach tasked domain experts to annotate 250 pieces from the Classical repertoire with four broad categories: *contentment*, *depression*, *exuberance*, and *anxiety*. The authors achieved a classification accuracy of 86.3% against expert ratings by training a Gaussian mixture model on acoustic and temporal features (Lu et al., 2006).

Another such study crowdsourced online annotations for 30-second excerpts from film scores. Importantly, each of the 200 songs was tagged by multiple listeners, 28.2 annotators on average, with one of eight mood categories: *sublime*, *sad*, *touching*, *easy*, *light*, *happy*, *exciting*, and *grand*. The authors empirically selected 29 acoustic features and trained a Support Vector Machine, reporting a cosine similarity of 0.73 between predicted and user-annotated labels (Wu and Jeng, 2008).

More recently, there have been efforts to develop mid-level features that may be better understood by a knowledgeable listener. These explainable features include perceptual concepts such as tonal stability, articulation, and rhythm. The authors trained a convolutional neural network using such features extracted from a set of 110 movie soundtracks to achieve a correlation of 0.71 compared to expert emotion annotations (Chowdhury et al., 2019). This approach encourages feature-importance analysis on these mid-level features, perhaps enabling future recommendation systems to provide context for its mood-based suggestions.

2.2 Incorporating Song Lyrics

Approaches using acoustic features alone have not yet proven entirely effective in predicating affective responses of music. The semantic gap between low-level audio features and human affective responses potentially limits the ability of systems using only raw acoustic information to predict emotional response to music (Panda et al., 2020). It is likely that music emotion prediction systems must be augmented with additional data in order to improve emotion recognition performance in any meaningful capacity (Yang and Chen, 2012).

Subsequently, researchers turned to a song's lyrics as a potential source of data to aid in the prediction of a song's emotional qualities. For 1000 pop songs, one study generated synthetic labels by comparing the similarity of Last.FM[2] tags to one of four mood category descriptors (*angry*, *happy*, *sad*, *relaxed*) using the WordNet[3] database. The authors then were able to predict mood categories with 62.5% accuracy using only lyrics, compared to their baseline accuracy of 89.8% using acoustic features. When the authors combined acoustic and lyric features, they improved classification accuracy to 92.4% (Laurier et al., 2008).

Another study reported that their lyric-only model (63.7%) outperformed its audio-based counterpart (57.9%) (Hu and Downie, 2010). A multimodal model using both feature sets marginally increased performance to 63.7% using a dataset covering 18 mood categories.

2.3 Direct Prediction from Lyrics

Recently, investigators have explored emotion recognition models based only on the text in the lyrics. One such study determined the valence and arousal values of individual words in the lyrics using established word lists. These values were then aggregated to create a song-level prediction of valence and arousal. The authors achieved a 74.25% classification accuracy relative to the All Music Guide[4] mood tags (Cano and Morisio, 2017).

Agrawal et al. applied a transformer approach based solely on song lyrics to achieve a 94.78% classification accuracy on a large dataset of lyrics and four emotion categories (Agrawal et al., 2021). This promising result demonstrates the ability to extract meaningful semantic information from music lyrics without the need for acoustic analysis.

[2] https://www.last.fm/
[3] https://wordnet.princeton.edu/
[4] https://www.allmusic.com/

2.4 Transformers

Transformers are deep learning models based on the principle of self-attention. This mechanism allows each token in an input to be weighted based on the context provided by surrounding tokens in order to capture an internal representation of the dependencies between elements. First introduced in 2017 (Vaswani et al., 2017), this architecture has proved especially successful with natural language processing tasks. More recently, transformer models have been adapted for emotion recognition of natural language (Chiorrini et al., 2021). Transformer models have also been recently applied in the area of music mood categorization, using lyrics as model input (Agrawal et al., 2021).

Bi-directional Encoder Representations from Transformers, or BERT (Devlin et al., 2019), is a popular transformer model for natural language understanding. This model comes pre-trained on a large dataset of English literature and Wikipedia articles. Although pre-training allows BERT-like models to be fine-tuned relatively quickly, training can still require immense compute resources, especially in the case of many NLP tasks where datasets can be quite large.

The RoBERTa model improves upon the original BERT model, adding additional model parameters and increasing the size of the training dataset by an order of magnitude (Liu et al., 2019). Although RoBERTa is able to exceed BERT's performance on many benchmark NLP tasks, this performance comes at the cost of significantly greater resources required for model training.

In an alternate approach, the DistilBERT model aims to lower the computational cost of training transformer models on large datasets by reducing the size of the model, and thereby significantly improving training and inference times (Sanh et al., 2019). The DistilBERT model reduces the number of model parameters by almost a factor of two while retaining competitive performance when compared to BERT.

In this work, we compare RoBERTa and DistilBERT models in the task of predicting emotion of songs directly from social media discussions.

3 Datasets

In this section we describe our selection of songs to consider in our experiment. We then detail our procedure for collecting musical discourse from social media platforms.

3.1 Music Emotion Datasets

Although music emotion recognition has been of particular interest in recent years in the field of music information retrieval, research is limited by a lack of available datasets suited for this task. These datasets require listeners to annotate musical excerpts. These experiments must be carefully controlled and usually occur in a lab environment or using an online crowdsourcing platform. Furthermore, these studies must employ large sample sizes, since the interpretation of music is highly subjective. We identified only four such datasets that provide continuous affective measurements in the valence-arousal space.

The DEAM dataset consists of 1,803 songs selected from royalty-free platforms and are songs likely unknown to the participant (Aljanaki and Soleymani, 2018). Listeners provided continuous annotations over the duration of the 45-second excerpt. Unlike other datasets, DEAM is able to provide the accompanying audio for each song, since these are not restricted by copyright. However, this also meant that these songs are relatively unknown. We were unable to consider this dataset because of an insufficient presence of social media commentary.

The Deezer dataset is a large set of 18,644 songs with synthetically generated affective annotations (Delbouys et al., 2018). These annotations were created with affective modeling based on the song's set of user tags on the website Last.FM. Although the large size of the Deezer dataset makes it a potentially valuable tool in this area of research, we exclude its consideration here as its emotion labels were estimated from natural language rather than human annotation.

In this work, we consider two datasets of songs with valence-arousal annotations. The first is the AMG1608 dataset, 1608 songs selected from the All Music Guide (Chen et al., 2015). In a crowdsourced task, listeners annotated 30-second excerpts. The study employed 665 annotators and achieved between 15 and 32 annotations per song. The second, the PmEmo dataset, contains annotations for 794 songs selected for their popularity on record industry charts (Zhang et al., 2018). The study recruited 457 undergraduate students to annotate 30-second excerpts. Because this study took place in a controlled lab setting, the authors also collected measurements of electrodermal activity.

Each of these datasets provides an artist name,

Dataset	Songs	Label Type	Scaling
AMG1608	1608	Crowdsourced	$[-1, 1]$
PmEmo	794	Lab Study	$[\ 0, 1\ ]$

Table 1: Comparison of the music emotion datasets

song title, and accompanying valence and arousal labels for each song. However, the scale used in each approach varies, as shown in Table 1. These differences reflect the differences in the methodology used in the data collection. We scale these values to $[-1, 1]$ for use in this study, but we concede that these differences limit the utility of cross-dataset comparisons. We show the distribution of valence and arousal labels as Figure 1.

From these two datasets, we extract the artist names and song titles to be used in our queries. In total, we consider 2402 songs, however duplicates were not removed, so that each dataset can be evaluated independently of one another.

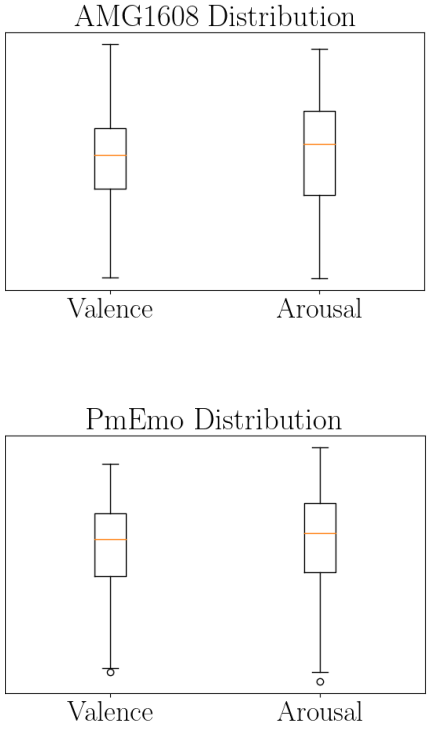


Figure 1: Box-and-whisker plots for the distribution of music emotion dataset labels in each dataset.

3.2 Social Media Data Collection

To explore the use of social media conversations as a feature space for music emotion prediction, we first must create a dataset of online discourse. We collect comment threads from social media for the songs featured in our two music emotion datasets: AMG1608 (Chen et al., 2015) and PmEmo (Zhang et al., 2018). Reddit, Twitter, and YouTube are large, popular social media platforms with active music subcultures, where individuals often converse about artists, songs, and concerts. In the case of YouTube, many use it as a platform to share and listen to music as well.

We collected our data over the course of two months in late 2021, harvesting all relevant commentary posted to date. For each song in our dataset, we query the platform for the artist name and track title. For YouTube and Reddit we extract the 10 highest rated submissions, based on likes and upvotes, respectively. We include all nested comments that appear as a response to a top-level comment. As a platform focused on short text posts, Twitter posts differ from the other two sites. Instead of a traditional reply chain, users retweet a post while potentially adding optional commentary. To avoid duplicating comments, we instead retrieve the top 100 tweets referencing the given song, excluding retweets. If a query for a song yields no submissions, we exclude that song from our dataset. In Table 2 we summarize our dataset of retrieved comments for each social media source.

Across both datasets, YouTube achieved the highest retrieval rates, finding more than 95% of the songs. This shows that YouTube supports a robust community for music-related discourse. The song retrieval rate for Reddit was also high, succeeding in finding at least 86% of the songs in either dataset. The retrieval rate for Twitter is notably lower, finding only 43% and 51% of songs in the two datasets. Because we could not find discussion of many songs, we conclude that Twitter is a less active medium to discuss opinions about music.

We show the distribution of retrieved comments as Figure 2. Overall we found more comments per song for songs in the PmEmo dataset across these three platforms than songs from AMG1608. This likely reflects the popularity of songs on record industry charts. We also observe that comments on Reddit tend to contain more words than those on YouTube, indicating that these discussions are frequently longer and perhaps more detailed than similar conversations on YouTube. As expected, given the 280-character limit, Twitter conversations are much shorter.

		Songs		Comments			Words	
		n	yield	n	μ	σ	μ	σ
AMG1608	Reddit	1431	89%	129,722	80.7	154.3	2400.8	69.1
	YouTube	1592	99%	217,093	135.0	57.7	2128.7	33.66
	Twitter	822	51%	5726	3.6	7.9	51.1	14.5
PmEmo	Reddit	627	86%	103,398	136.6	218.7	3810.5	56.8
	YouTube	730	95%	121,546	160.6	63.9	2172.1	44.1
	Twitter	331	43%	2699	3.6	7.3	46.0	15.2

Table 2: Summary statistics describing our dataset of social media commentary by social media source.

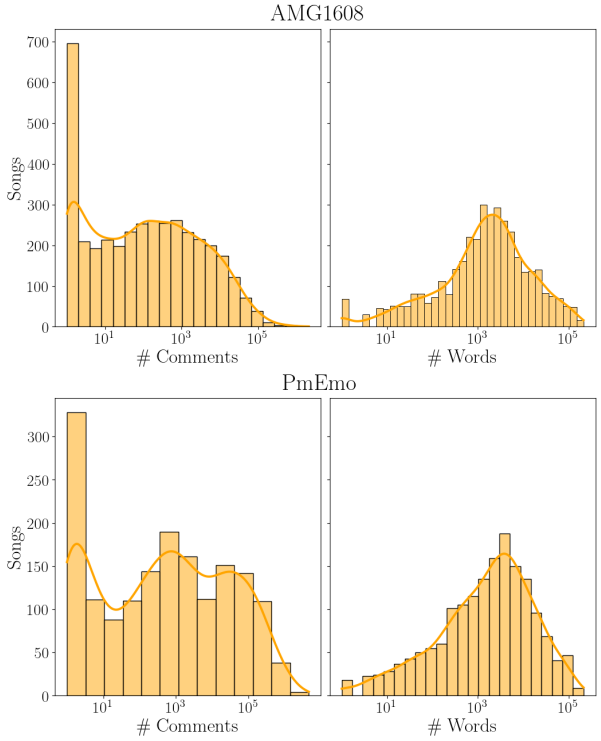


Figure 2: Comment and word distributions of our discourse datasets.

4 Experiment and Results

To test the utility of social media discourse towards the prediction of emotion in music, we conduct a deep learning experiment. We compare two powerful pre-trained transformer models for natural language understanding – DistilBERT (Sanh et al., 2019) and RoBERTa (Liu et al., 2019). In this section, we describe our model architecture, explain our experimental design, and present our results.

4.1 Model Architecture

Our model architecture consists of a pre-trained transformer model augmented with a densely connected neural network to predict regression targets from the last hidden state of the transformer, referred to as the regression head. We use the `TFDistilBertBase` and `TFRobertaBase` model implementations provided by the Huggingface deep natural language processing library[5].

Each input to the model consists of one text comment, with corresponding music valence and arousal labels. Sentences are tokenized using Huggingface's `TokenizerFast` library. We use the default input size of 128. Comments longer than 128 words will be truncated, and comments shorter than this sequence length will be right-padded with 0-tokens. For each model, we use the default language model architecture: six layers and twelve self-attention heads in the case of DistilBERT, and twelve layers with twelve self-attention heads for RoBERTa.

Deep learning models, such as transformers, naturally support multi-target regression through the use of a set of output nodes. This approach allows our model to predict valence and arousal as co-dependent values instead of independent labels as has often been done in prior approaches. As a regression head for each pre-trained transformer model, we append two fully connected layers and an output layer of two nodes, representing valence and arousal. We use mean-squared error as our loss function and a learning rate of 1×10^{-5}.

4.2 Experimental Design

We randomly partition our dataset into training (0.70), validation (0.15), and test (0.15) sets. We split our dataset at the song-level, rather than the comment-level to prevent potential information leakage from our test set. Valence and arousal labels are normalized and scaled to [0, 1].

We filtered the raw social media comments to remove URLs and HTML tags. Since transformer

[5] https://huggingface.co/models

models are pre-trained on large corpora of English text, they expect the input to adhere to standard grammatical structure. Therefore we do not filter stop-words or words with neutral sentiment.

The model outputs consist of a valence and arousal prediction for each comment. To aggregate a prediction at the song-level, we take the mean of the predictions across all comments for a particular song. We evaluate our models' performance using the Pearson's correlation between our predictions and ground truth values for valence and arousal.

4.3 Results

We begin by comparing the RoBERTa and DistilBERT models. We train each model using the combined social media commentary from Reddit, Twitter, and YouTube. We considered any song in the AMG1608 and PmEmo datasets as long as they are included in at least one of the three social media sources.

4.3.1 Model Comparison

BERT-like models are known to require minimal additional task-specific training due to their pre-trained nature (Devlin et al., 2019). In preliminary experiments we trained each model for 10 epochs and observed that our models converged between one and three epochs, depending on the dataset. In order to compare the performance of these two models while controlling for overfitting, we train each model for two epochs. We compare the results for each model and dataset in Table 3.

		DistilBERT	RoBERTa
AMG1608	Valence	0.49	0.51
	Arousal	0.64	0.63
PmEmo	Valence	0.72	0.71
	Arousal	0.64	0.64

Table 3: Comparison of DistilBERT and RoBERTa performance after two epochs of training.

The performance between the two models is comparable and the differences are not statistically significant. However, the difference between computational cost for these models is considerable. In our experiments, the DistilBERT model completed training in less than half the runtime as RoBERTa. Because the models are comparable in predictive performance, we use DistilBERT in our subsequent experiments.

4.3.2 Source Comparison

Next, we train individual models for each social media source, Reddit, Twitter, and YouTube. Because not every song was found on each platform, the number of songs used to train each model varies (see Table 1). We compare these three source-specific models with another model that combines comments from all three sources. We show the results of this experiment for the AMG1608 song list using DistilBERT as Table 4.

AMG1608

	Reddit	Twitter	YouTube	All
Valence	0.32	0.23	0.62	0.49
Arousal	0.56	0.34	0.72	0.64

PMEmo

	Reddit	Twitter	YouTube	All
Valence	0.56	0.26	0.68	0.72
Arousal	0.60	0.16	0.52	0.66

Table 4: Results of DistilBERT trained for two epochs for each social media source and song list.

We observe the highest overall performance on the YouTube subset achieving valence and arousal correlations of 0.62 and 0.72, respectively, on the AMG1608 dataset. Across both datasets the YouTube model tends to outperform the Reddit model, even though both data sources contain a comparably large number of comments. Although we matched fewer songs and collected fewer comments from Twitter, we still observe weak correlations between Twitter discourse and the song's valence and arousal annotations.

In addition to the source-specific models, we also examined the performance of the combined model. For the AMG1608 song list, the combined model demonstrated improvement over either the Reddit or Twitter model alone. However, aggregating all sources together reduced performance compared to the YouTube model alone by 21% for valence and 11% for arousal. Conversely, we observe the combined model outperformed any of the three source-specific models for the PMEmo songs.

Ultimately, these conflicting trends likely reflect the differences between the specific songs present in the two datasets, rather than differences in utility between the social media sources. This again underscores the field's need for much larger and

diverse datasets of music annotated with human affective responses.

5 Discussion

In this work, we present a novel approach to estimate the emotional qualities of a song solely through analysis of discussion of that song on social media. We create large datasets of conversations discussing music from three social media platforms. We train natural language transformer models to predict affective measurements of the song, directly from this discourse alone. Overall, we observe moderate correlations between our predictions across the three social media sources. These results indicate that the semantic information embedded in these comments can potentially be used to help predict affective responses to music.

5.1 Limitations and Future Work

We found that the distributions of our model's predictions tend to cluster closely to the center of the valence-arousal space. We show the distribution of our predictions and the actual value as Table 3.

We hypothesize that this occurs for two reasons. First, our unfiltered data may be too noisy for meaningful sentiment analysis at scale without some initial filtering of the comments. As future work, we will investigate approaches to clean the dataset while managing selection bias. We will consider dropping comments that may have adverse effects on our model performance, such as those of an insufficient length, those containing a low or negative score, or those generated by bots. We will also investigate dropping any comments which do not contain an affective word, using established affective word lists.

Secondly, we predict values for each comment and aggregate these comment-level predictions to estimate a value for the entire song. This approach was convenient to facilitate our exploration of existing pre-trained transformer architectures for this task. However, this aggregation risks losing valuable semantic information. For example, the sentiment contained within one comment may be cancelled by another with opposing sentiment, reducing them to an average neutral sentiment. As future work, we will investigate model architectures which may allow us to better retain inter-comment dependencies, such as Relation-Aware Transformers (Wang et al., 2021). We will also explore new architectures that support longer input sequences, such as the `xl-net` model (Yang et al., 2020) that has been recently applied to music mood classification from lyric analysis (Agrawal et al., 2021).

Our model is bounded by the requirement of a sufficient corpus of social media conversation pertaining to a song. This restricts this approach's efficacy in cases of newly released music or niche genres. An acoustic or lyrical approach, in contrast, would handle these scenarios equally to more popular song examples. In future experiments we will compare the performance of multimodal music emotion recognition systems when augmented with a social media input.

All our models trained exclusively on Twitter data performed poorly compared to the other source-specific models. However, our data collection method differed between social media platforms. We intend to repeat our Twitter data collection process in order to retrieve far more comments than available in this work. Also, we will revise our data-mining approach to include responses while explicitly filtering out reduplicated text. Despite these improvements, the combination of comment length restrictions and low yield rates for mentions of a song on the platform lead us to anticipate finding less available data on Twitter compared to YouTube or Reddit.

Additionally, we will explore other potential sources for social media commentary. For example, the community annotated tags on the site Last.FM have been used to generate features for music emotion recognition (Bischoff et al., 2009; Delbouys et al., 2018). Last.FM has recently added "Shouts", which allows users to post free-form comments in response to a song. To our knowledge no existing work has attempted to use sentiment analysis on Last.FM conversations for music emotion prediction. We will investigate commentary on the website SoundCloud[6] as well. SoundCloud is unique in that it associates posts with specific timestamps in the recording. This temporal information could be useful in determining changes of sentiment over the course of a piece of music. As we continue to collect additional data, we plan to make our dataset publicly accessible to facilitate further research into the use of social media commentary for music emotion recognition.

[6] https://soundcloud.com/

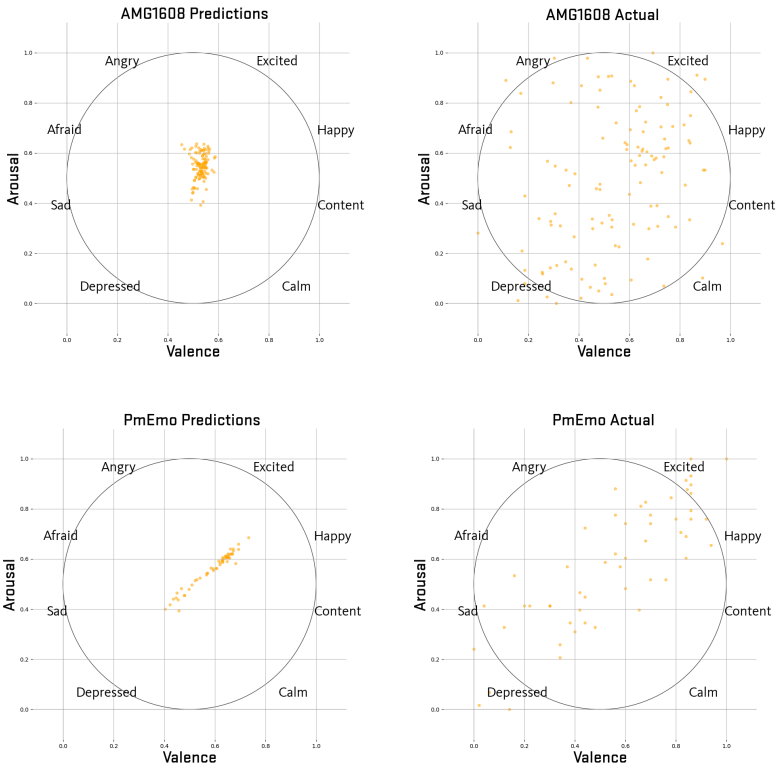


Figure 3: Distribution of DistilBERT predictions on songs in our test set for AMG1608 and PmEmo.

5.2 Contributions

In this work we assess the feasibility of predicting music emotion indirectly from social media discourse. By leveraging freely available social media commentary, we explore alternate modalities to make determinations about a musical affect when the raw waveform may not be available.

We create a novel dataset of conversations related to music from Reddit, YouTube, and Twitter. These comments correspond to the songs annotated in two music emotion datasets frequently cited in the literature. This correspondence allows comparison of the results of our supervised deep learning approach with human annotated labels of musical affect as well as to existing audio-based methods for estimation of musical affect.

Our results demonstrate that the conversations from social media platforms like Reddit, YouTube, and Twitter do contain semantic information which may be relevant to the task of music affect prediction. Although these correlations are moderate at best, they show the potential utility of this approach in music emotion recognition, especially if employed in a multimodal system that also compares audio and lyric-derived features. To our knowledge, this is the first approach to predict valence and arousal of musical songs using only conversational information from social media platforms.

5.3 Conclusion

Research investigating the automatic detection of the emotional qualities of music is often hindered by the absence of large-scale datasets annotated with affective responses to music. Such datasets are difficult to create and copyright concerns often limit the release of the raw audio needed by many machine learning approaches. Motivated by the use of song lyrics to predict emotion in music, in this work we explore the novel task of leveraging social media discourse to predict affective responses to music. We trained natural language transformer models using only discourse from three social media sites to predict the valence and arousal of pieces of music. We found moderate correlations between discourse about songs on Reddit, Twitter, and YouTube and the human annotated values of affective responses to those songs. Therefore, it is possible to predict the affective qualities of some songs directly from online conversations.

References

Yudhik Agrawal, Ramaguru Guru Ravi Shanker, and Vinoo Alluri. 2021. Transformer-based approach towards music emotion recognition from lyrics. *Advances in Information Retrieval. ECIR 2021*, 12657:167–175.

Anna Aljanaki and Mohammad Soleymani. 2018. A data-driven approach to mid-level perceptual musical feature modeling. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, ISMIR, pages 615–621.

Kerstin Bischoff, Claudiu S Firan, Raluca Paiu, Wolfgang Nejdl, Cyril Laurier, and Mohamed Sordo. 2009. Music mood and theme classification - a hybrid approach. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, ISMIR, pages 657–662.

Erion Cano and Maurizio Morisio. 2017. Moodylyrics: A sentiment annotated lyrics dataset. In *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, page 118–124. ACM.

Yu-An Chen, Yi-Hsuan Yang, Ju-Chiang Wang, and Homer Chen. 2015. The AMG1608 dataset for music emotion recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, page 693–697. IEEE.

Andrea Chiorrini, Alex Mircoli, Claudia Diamantini, and Domenico Potena. 2021. Emotion and sentiment analysis of tweets using BERT. In *Proceedings of the EDBT/ICDT 2021 Joint Conference*.

Shreyan Chowdhury, Andreu Vall, Verena Haunschmid, and Gerhard Widmer. 2019. Towards explainable music emotion recognition: The route via mid-level features. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*, ISMIR, pages 237–243.

Remi Delbouys, Romain Hennequin, Francesco Piccoli, Jimena Royo-Letelier, and Manuel Moussallam. 2018. Music mood detection based on audio and lyrics with deep neural net. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, ISMIR, pages 370–375.

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, pages 4171 – 4186.

Xiao Hu and J. Stephen Downie. 2010. Improving mood classification in music digital libraries by combining lyrics and audio. In *Proceedings of the 10th annual joint conference on Digital libraries*, page 159–168. ACM.

Cyril Laurier, Jens Grivolla, and Perfecto Herrera. 2008. Multimodal music mood classification using audio and lyrics. In *2008 Seventh International Conference on Machine Learning and Applications*, page 688–693.

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:1907.11692*, 1907(1907.11692).

Lie Lu, D. Liu, and Hong-Jiang Zhang. 2006. Automatic mood detection and tracking of music audio signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):5–18.

Renato Panda, Ricardo Manuel Malheiro, and Rui Pedro Paiva. 2020. Audio features for music emotion recognition: a survey. *IEEE Transactions on Affective Computing*, pages 1–20.

James A. Russell. 1980. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178.

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

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*, NIPS'17, page 6000–6010.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2021. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv:1911.04942*.

Tien-Lin Wu and Shyh-Kang Jeng. 2008. Probabilistic estimation of a novel music emotion model. In *Proceedings of the 14th international conference on Advances in multimedia modeling*, MMM'08, page 487–497. Springer-Verlag.

Yi-Hsuan Yang and Homer H. Chen. 2012. Machine recognition of music emotion: A review. *ACM Transactions on Intelligent Systems and Technology*, 3(3):1–30.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Kejun Zhang, Hui Zhang, Simeng Li, Changyuan Yang, and Lingyun Sun. 2018. The PMEmo dataset for music emotion recognition. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, page 135–142. ACM.

Customer Sentiments Toward Saudi Banks During the Covid-19 Pandemic

Dhuha Alqahtani, Lama Alzahrani, Maram Bahareth, Nora Alshameri, Hend Al-Khalifa, Luluh Aldhubayi
Information Technology Department, College of Computer and Information Sciences
King Saud University
Riyadh, Saudi Arabia
{441203741, 442203072, 439203913, 442204277}@student.ksu.edu.sa,
{hendk, laldubaie}@ksu.edu.sa

Abstract

In view of the recent interest of Saudi banks in customers' opinions through social media, our research aims to capture the sentiments of bank users on Twitter. Thus, we collected and manually annotated more than 12,000 Saudi dialect tweets, and then we conducted experiments on machine learning models including: Support Vector Machine (SVM), Random Forest (RF), and Logistic Regression (RL) as well as state-of-the-art language models (i.e. MarBERT) to provide baselines. Results show that the accuracy in SVM, LR, RF, and MarBERT achieved 82.4%, 82%,81%, and 82.1% respectively. Our models code and dataset will be made publicly available on GitHub.

1 Introduction

Over the last decade, social media sites generated enormous content online which conducts challenges to decision making and manual content analysis (Almuqren & Cristea, 2021). The Saudi banks sector has undergone essential changes over the decade. They have taken advantage of expanding their operations of product diversification as well as the features of scale and scope economies. The changes affected the feelings and sentiments of customers and their dealings with banks as well as the choice of the bank based on the features and services they provide. In addition, appropriate treatment of the customer is a milestone in the selection of a bank, and good treatment is characterized by quick interaction, offers and satisfactory customer service.

Social media sites are one of the platforms where customers' opinions can be collected and analyzed. Our research aims at capturing customers' sentiments of Saudi banks, by analyzing their feelings and revealing their opinions.

Although there are several banks in Saudi Arabia, yet, for the purpose of this study, only four Saudi banks were selected namely (AlRajhi bank, Alinma bank, Saudi National Bank (SNB), and Saudi Investment Bank (SAIB)) as they are considered the top Saudi banks according to Semrush website[1].

We collected a corpus of more than 12,000 Arabic tweets and manually labeled them with three sentiments (positive, negative, and neutral). Then we applied three machine learning models on the corpus, namely: Support Vector Machine (SVM), Logistic Regression (LR), and Random Forest (RF) as well as MarBERT pretrained model for sentiment analysis.

The rest of this paper is organized as follows. Section 2 presents the state-of-the-art and related work on banking sentiment analysis. Section 3 demonstrates the corpus construction steps and the annotation process. Section 4 presents the analysis results which include model selection and corpus evaluation. Section 5 concludes the paper with discussion of the results, limitations and future work.

[1] https://www.semrush.com/website/top/saudi-arabia/banking/

2 Related work

One of the research areas in sentiment analysis is to capture customer sentiments regarding vital services such as Banking, by knowing and classifying customer opinions, this will help improve the titled services.

In this section, we will highlight previous work in sentiment analysis for banks. The study by Kazmaier and Vuuren (2020) conducted a sentiment analysis as unstructured customer reviews that related to services and products for a retail bank in South Africa. They used a machine learning model to detect sentiment with a high level of qualified performance. The result shows that custom learning-based models are better than previous models that used commercial tools and were pre-trained for sentiment classification.

Eksa Permana et al. (2020) present a study to determine the customer sentiment on mobile banking applications to specify the aspects that need to be maintained or improved
in the application. They used Naive Bayes models to discover the sentiment analysis. The results displayed high accuracy at the value of k=5, which is accuracy with a value of 86.762% and precision with 92.482% also 93.474% for recall.

(Gavval et al., 2019) proposed a visual sentiment analysis for customer complaints related to services and products of four superior Indian banks. The author leverages the available bank's compliant responses dataset which consists of 749k consumer opinion on four Indian banks namely: Axis Bank, HDFC Bank, ICICI Bank, and SBI. They used a Self-organizing feature map (SOM) and CUDA-based Self Organizing Feature Map (CUDASOM) algorithm. They mentioned that the performance of CUDASOM algorithm increased the speed of CPU up to 44 times which improved the result of experiment.

Krishna et al. (2019) applied sentiment analysis on Indian bank's customer complaints. They used machine learning techniques such as Support vector machines (SVM), Naive Bayes (NB), Logistic Regression (LR), Decision Tree (DT), and Random Forest (RF) towards applying preprocessing the raw textual data. The findings showed that the ML models with three banks dataset performed the best result where LR obtained (77%), RF (77%), SVM (75%), and NB (74%).

From the previous work, we noticed that several research used machine learning methods such as (Kazmaier and Vuuren, 2020), (Eksa Permana et al.,2020), (Gavval et al., 2019), and (Krishna et al., 2019).

As shown in Table 1, we present some of studies with the name of the bank or region, the applied model and results. In our research, the main contribution is to construct a gold standard dataset for the Saudi banks customers' sentiment.

Finally, the corpus will be evaluated using machine learning and pretrained models.

Study	Bank	Method	Result
Kazmaier and Vuuren (2020)	Retail bank in South African	Naive Bayes, SVM, CNN, ANN, and LR	LR: 84.00%
Eksa Permana et al. (2020)	Mobile banking applications	Naive Bayes	NB:86.76%
Gavval et al. (2019)	Four superior Indian banks	CUDASOM algorithm	Speed up the CPU performance 44 times
Krishna et al. (2019)	Indian bank	SVM, NB, LR, DT, RF	LR: (77%), RF: (77%), SVM:(75%), NB: (74%)

Table 1: Summary of relevant studies

3 Dataset

This section contains two parts related to constructing the Saudi Bank Corpus. The first part is to collect the data and a description of the corpus. While the second one represents the corpus annotation phase and statistics about the corpus.

3.1 Data Collection

The Saudi banks' corpus data were collected from Twitter using Tweepy python library. We used the banks' Twitter account mentions of the four Saudi banks to retrieve the tweets.

The data were collected for a whole month, starting from the 1st of September 2021 until the 30th of September 2021, throughout the COVID-19 outbreak when all bank institutions switched their services online. During that period, most people encountered online issues when using their bank services. Therefore, due to the COVID-19 constraints, social media platform such as Twitter was the main channel to communicate with the bank's organizations. The retrieved tweets were focusing on getting customers' feedback or opinion on the banks. A total of 52,254 tweets were collected during the whole month.

As we scrapped the tweets using the official bank account (@bank_account) on Twitter, as a result, the number of tweets crawled was depend on the number of bank customers, whether the Twitter account is active or not, and if there is another account for customer complaints. Moreover, the bank that has a separate customer care account received too many tweets from their customers than the bank that has one official account. Moreover, the bank that is 24/7 active and responds quickly has many user replies and comments. The special occasions such as National day and the recent collaboration between banks plays a crucial role in increasing customer response accordingly.

3.2 Data Preprocessing

Since the data is collected from Twitter, it could contain noises and unwanted symbols, which could eventually affect the model's performance. In this section, several normalization and cleaning steps were applied on the collected data using regular expression patterns. The pre-processing steps are listed below:

- **Normalizing Letters:** Some of the letters got changed to have fixed shape for example "إأآ" would be "ا".
- **Normalizing numbers:** All the numbers have standard representation in our case, we used Arabic numbers for example "٩٩٩" would be "9".
- **Remove duplicate letters:** All the letters that occur for more than two times get limited to two times since some words could have the same letter twice, this process has a special case where the laugh such as "هههههه" replaced by "ضحك"
- **Remove duplicate tweets:** Any duplicated tweet is kept only once to make sure we don't annotate the same sentence more than once.
- **Removing punctuation marks:** All the punctuation marks got deleted except the "!?." Since it could have meaning for the sentiment.
- **Removing duplicate whitespace:** White spaces between words were eliminated to one space only.
- **Remove hyperlinks or URLs:** This process consists of removing all the URLs (HTTP or HTTPS).
- **Remove hashtags and mentions:** For example, #something" and "@someone" will be deleted.
- **Remove special characters:** Special characters include ".,$,%,&,*, etc." were deleted.
- **Remove Arabic diacritics:** Deleting Tashkeel including "ً ٌ ٍ َ ُ ِ ّ ْ"
- **Remove Tatweel words:** Refers to removing the stretching word space that is represented as (-) symbol.
- **Remove non-Arabic word**s: Any non-Arabic words got deleted, for example, English words will get deleted.
- **Remove sentences with less than five words:** Sentences with less than five words could be too short to represent any type of sentiment so, we deleted them.
- **Eliminate tweets with neutral sentiment:** we used Camel and Mazajak tools that is designed for Arabic SA. We used it to eliminate the neutral tweets, since we wanted to make sure that the tweets would contain positive or negative sentiment.

3.3 Data annotation

Based on the type of analysis that will be performed on the corpus "sentiment analysis," it was decided that the annotation will be a single level annotation, where each sentence will be annotated into either positive, negative, or neutral. In order to annotate the data, the data was split equally into two divisions. Specifically, each division was annotated by two annotators from the authors. Furthermore, the annotation was made blindly, which means that each annotator did not have access to the other annotators' annotation. At the end, the opinion of the two annotators got compared. If there was no agreement on the label, the tweet was eliminated.

The manual annotation process required quality assurance since human opinion varied in nature according to several aspects such as education, age,

culture, and more. Therefore, it is important to measure the level of agreement between different annotators. The inter-annotator agreement (IAA) is a measure that is used if there were multiple annotators who selected the same decision for a particular category. The IAA is used to validate and interpret the labeling result. Three common metrics are used the percent agreement, Cohen kappa, Fleiss' kappa to compute the IAA for the annotation process. Percent agreement is the simplest method, but it doesn't consider the possibility of random guesses annotation. Cohen kappa and Fleiss' kappa solve the drawbacks of Percent agreement. Cohen kappa is more suitable for two annotators while Fleiss' kappa is suitable for more than two annotators. So, the selection of the metrics is according to the number of individuals who annotate the corpus. Therefore, We used the Cohn kappa metric in this research since two annotators annotated each record. The metric obtains a result above 0.65, which consider as a substantial agreement and it means that the result of the annotation is suitable to train the classification models.

The final corpus was saved as a CSV format that contains five columns as shown in figure 1. the first column shows the cleaned text. the second column contain the name of the bank either AlRajhi, Alinma, Saudi Investment bank (SAIB) or the Saudi National banks (SNB). The third and fourth columns contain the tokenized words of each sentence, one with stop words and the other without stop words. The last column has the annotation label made by the annotators of the tweet. After the corpus pre-processing and annotation, we performed some statistical analysis on the corpus. The analysis shows that our corpus contains 1,567,628 tokens and 82,015 word types. In addition, according to the bank names: we gathered 7,775 tweets for AlRajhi bank, 2,705 for SNB, 920 for Alinma bank, and 648 for SAIB. Moreover, the results of the corpus annotations show that among the gathered tweets, 8,669 of them were labeled as "Negative", 2,143 were labeled as "Positive", and 1,236 tweets were labeled as "Neutral".

	A	B	C	D	E	F	G
1	Tweet	Bank	Tokens	Tokens without stop words	Annotator 1 & 3	Annotator 2 & 4	Final annotation
2	الله يقلعك من بنك دمر مستقبلنا الله حسبنا	SAIB	['الله', 'يقلعك', 'من', 'بنك', 'دمر', 'مستقبلنا', 'الله', 'حسبنا']	['الله', 'يقلعك', 'بنك', 'دمر', 'مستقبلنا', 'الله', 'حسبنا']	NEG	NEG	NEG
3	صار لي يومين بحاول اوصل لاي شخص من	SAIB	['صار', 'لي', 'يومين', 'بحاول', 'اوصل', 'لاي', 'شخص', 'من']	['صار', 'يومين', 'بحاول', 'اوصل', 'شغال', 'اوصل']	NEG	NEG	NEG
4	صادق كثير من الزبائن يشتكون منه الصراحه	SAIB	['صادق', 'كثير', 'من', 'الزبائن', 'يشتكون', 'منه', 'الصراحه']	['صادق', 'كثير', 'زبائن', 'يشتكون', 'منه', 'صراحه']	NEG	NEG	NEG
5	ماقدرنا ناخذ بطاقه مدي وحسابي ذهبي وهو	SAIB	['ماقدرنا', 'ناخذ', 'بطاقه', 'مدي', 'وحسابي', 'ذهبي', 'وهو']	['ماقدرنا', 'ناخذ', 'بطاقه', 'مدي', 'وحسابي', 'ذهبي']	NEG	NEG	NEG
6	الافضل ، خلل ، كبير ، عندكم	SAIB	['الافضل', 'خلل', 'كبير', 'عندكم']	['افضل', 'خلل', 'كبير', 'عندكم']	NEG	NEG	NEG
7	وراتبهم نصبا الساط وش يدور من عروض به	SAIB	['وراتبهم', 'نصبا', 'الساط', 'وش', 'يدور', 'من', 'عروض', 'به']	['راتبهم', 'نصبا', 'الساط', 'وش', 'يدور', 'عروض', 'بعد']	NEG	NEG	NEG
8	اما كذا وان التجير على دقيقه واثنت دقيقه عا	SAIB	['اما', 'كذا', 'وان', 'التجير', 'على', 'دقيقه', 'واثنت', 'دقيقه', 'عا']	['كذا', 'التجير', 'دقيقه', 'واثنت', 'دقيقه']	NEG	NEG	NEG
9	لوسمحتوا قاعدين تتصل عليكم محد يرد عنا	SAIB	['لوسمحتوا', 'قاعدين', 'تتصل', 'عليكم', 'محد', 'يرد', 'عنا']	['لوسمحتوا', 'قاعدين', 'تتصل', 'عليكم', 'محد', 'يرد', 'استفسار']	NEG	NEG	NEG
10	لاتورطونون مثلي البنوك الثانيه افضل من واقع	SAIB	['لاتورطونون', 'مثلي', 'البنوك', 'الثانيه', 'افضل', 'من', 'واقع']	['لاتورطونون', 'مثلي', 'البنوك', 'الثانيه', 'افضل', 'واقع', 'خيره']	NEG	NEG	NEG
11	انا معهم وللازلت اسدد طيله الست سنوات الا	SAIB	['انا', 'معهم', 'وللازلت', 'اسدد', 'طيله', 'الست', 'سنوات', 'الا']	['انا', 'معهم', 'للازلت', 'اسدد', 'طيله', 'الست', 'سنوات']	NEG	NEG	NEG
12	احسنت ربي يسهلك الخير انا ماخذ من عندهم	SAIB	['احسنت', 'ربي', 'يسهلك', 'الخير', 'انا', 'ماخذ', 'من', 'عندهم']	['احسنت', 'ربي', 'يسهلك', 'الخير', 'انا', 'ماخذ', 'عندهم']	NEG	NEG	NEG
13	الشركات الكبيره واني العملاء طق مقاتلي عا	SAIB	['الشركات', 'الكبيره', 'واني', 'العملاء', 'طق', 'مقاتلي', 'عا']	['الشركات', 'الكبيره', 'واني', 'العملاء', 'طق', 'مقاتلي', 'عا']	NEG	NEG	NEG
14	ما انسى تجربه اخوي معاكم ووصلكم بالبنك ام	SAIB	['ما', 'انسى', 'تجربه', 'اخوي', 'معاكم', 'ووصلكم', 'بالبنك', 'ام']	['انسى', 'تجربه', 'اخوي', 'معاكم', 'ووصلكم', 'البنك']	NEG	NEG	NEG
15	اي والله وممكنه التعامل كذا	SAIB	['اي', 'والله', 'وممكنه', 'التعامل', 'كذا']	['اي', 'والله', 'وممكنه', 'التعامل', 'كذا']	NEG	NEG	NEG
16	هذا البنك على اساس من الرحمه الي عندهم	SAIB	['هذا', 'البنك', 'على', 'اساس', 'من', 'الرحمه', 'الي', 'عندهم']	['البنك', 'اساس', 'الرحمه', 'الي', 'عندهم']	NEG	NEG	NEG
17	ما ادري يدرون بكره ولاكيف	SAIB	['ما', 'ادري', 'يدرون', 'بكره', 'ولاكيف']	['ادري', 'يدرون', 'بكره', 'كيف']	NEG	NEG	NEG
18	مو معقوله انتظر دقيقه على الجوال ولي اجي	SAIB	['مو', 'معقوله', 'انتظر', 'دقيقه', 'على', 'الجوال', 'ولي', 'اجي']	['مو', 'معقوله', 'انتظر', 'دقيقه', 'جوال', 'اجي']	NEG	NEG	NEG
19	كفيه حق بنك البنك رقي في التعامل متقهرين	SAIB	['كفيه', 'حق', 'بنك', 'البنك', 'رقي', 'في', 'التعامل', 'متقهرين']	['كفيه', 'حق', 'بنك', 'البنك', 'رقي', 'التعامل', 'متقهرين']	POS	POS	POS
20	للاسف تم احتسابها في بطريقه خاطء كيف	SAIB	['للاسف', 'تم', 'احتسابها', 'في', 'بطريقه', 'خاطء', 'كيف']	['للاسف', 'احتسابها', 'بطريقه', 'خاطء', 'كيف']	NEG	NEG	NEG
21	مو طبيعي الانتظار في فلكس كول جلست اتنا	SAIB	['مو', 'طبيعي', 'الانتظار', 'في', 'فلكس', 'كول', 'جلست', 'اتنا']	['مو', 'طبيعي', 'انتظار', 'فلكس', 'كول', 'الخدمه']	NEG	NEG	NEG

Figure 1: Sample of the dataset file

4 Experiments and Results

4.1 Model Selection

The second stage in conducting the sentiment analysis on Saudi banks was the model selection. We had many experiments in order to find the best model that gains the highest accuracy in detecting sentiments. These experiments involved implementing three machine learning techniques which are: Support vector machine (SVM), Logistic regression (LR), Random Forest (RF). In addition, we implemented a fourth model using MarBERT model. The feature representation was implemented using the term frequency-inverse document frequency (TF-IDF). According to (Al-Twairesh et al., 2017), the TF-IDF representation shows a good performance on Arabic text. To train the model, we split the corpus into 80% for training and 20%for testing. The decision for selecting these models was based on the Arabic research community in which the result of the selected model showed good predictions especially for sentiment analysis. Different appropriate parameters were used for each model to obtain the desired performance.

The first machine learning model is the Support Vector Machine (SVM) which is a supervised machine learning algorithm. The second machine learning model is the logistic regression (LR) which is a supervised machine learning algorithm and used widely for binary classification problems. However, LR can be used to perform a multilabel classification task such as sentiment analysis. In the current work, we evaluated the corpus with multilabel logistic regression classification for sentiment analysis namely positive, negative, and neutral. The logistic regression has shown a successful result in sentiment analysis in diverse languages such as English and Arabic (Bessou and Aberkane, 2019). Thus, motivated by the promised result achieved in the previous study, we will

conduct a corpus evaluation using logistic regression and perform fine-tuning with regularization parameters using a grid search to obtain the best result. The model has been evaluated with the predefined parameters using 5-fold cross-validation. The logistic regression model has shown improvement in performance with 82% accuracy.

The third machine learning model is the Random forests algorithm. The Random forests algorithm is a supervised machine learning algorithm that builds many individual decision tree classifiers. The main advantage of using the Random forests algorithm is that the generated decision trees are not correlated, therefore the classification error made by one tree is not seen by the other decision trees. On the other hand, random forest algorithm suffers from the slowness disadvantage. We apply this algorithm by using Random Forest Classifier module, Randomized SearchCV() method, and 5-fold cross-validation. Randomized SearchCV() apply tunning on the parameters to choose randomly the optimal parameters.

The last experiment was conducted using the MarBERT model, which is a pre-trained language model based on the Arabic dialectical data from social media. We fine-tuned the model based on the model provided by Abdul-Mageed et al. on GitHub page[2]. The model hyperparameters were selected based on the original model where the learning rate equals to 2×10-6, the batch size equals to 32, the maximum sequence length equals to 128, and the epoch equals to 5.

4.2 Model Evaluation

For evaluating the models' performance, the following metrics were used: Accuracy, Precision, Recall, and F-measure.

For all classifiers, we have computed the TF-IDF method which plays a crucial role in the training stage to select the most important words among the dataset.

Table2 shows the performance of all models using the metrics: accuracy, precision, recall, and F1 score. Overall, the evaluation results were quite close among all models. However, the best performing result was achieved by SVM with 82.4% accuracy and an F1 score of 79%. On the contrary, logistic regression reported the most stable results in accuracy and F1 score with 82% and 81% which represent the highest result compared with other classifiers.

Model name	Accuracy	Precision	Re-call	F-measure
MarBERT	82.1	70.2	66.3	68.0
Logistic regression (LR)	82.0	**80.0**	**82.0**	**81.0**
Random Forest (RF)	81.0	78.0	81.0	77.0
Support Vector Machine (SVM)	**82.4**	**80.0**	**82.0**	79.0

Table 2: models' results

4.3 Error analysis

To demonstrate the model accuracy and justify the performance of the results. An error analysis process was performed on our best used model which is LR model; to show where the model succeeds and failed to classify the data and set observation for the model limitations.

Figure 2 shows the confusion matrix of the classification error rate for each category (sentiment labels) for the LR model. After walking through the confusion matrix for the model, it is clearly noticeable what categories were misclassified and identified errors made by the classifiers. To demonstrate more, we grouped all these errors and created an observation for common errors in the prediction against the original dataset. We observed that the most misclassified class is NEU label, it is mostly misclassified as NEG. We found this case 158 times out of 244 NEU label tweets. This represent 64% of the actual NEU data that means that most of the NEU data has been misclassified, we attributed the reason of the misclassification to the imbalanced dataset. The second most error was POS label misclassified as NEG; it occurred 133 times which represent 29.7% of the POS tweets. The least error cases were for the NEG where nit

[2] GitHub - UBC-NLP/marbert: UBC ARBERT and MarBERT Deep Bidirectional Transformers for Arabic

was misclassified as NEU; it appeared only 50 times which represent 2.9% of NEG tweets.

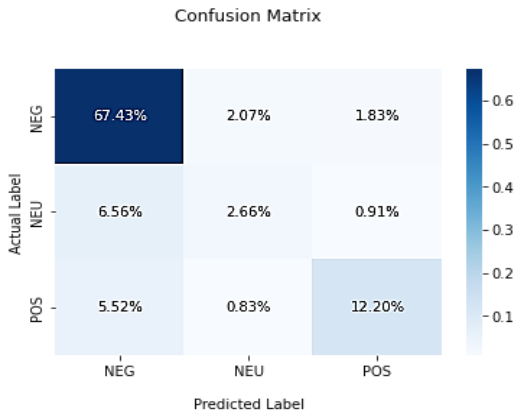


Figure 2: Confusion Matrix of LR model with percentage.

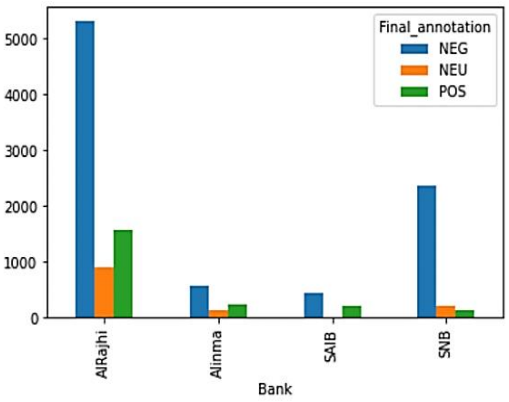


Figure 3: Distribution of tweets according to polarity: Positive, Negative, Neutral per bank.

We conclude that the misclassification errors occur as a result of imbalanced dataset. We noticed that because the NEG class represents about more than 70% of the dataset, that leads to misclassification of POS and NEU as NEG label. To deal with this issue we are using F-measure for evaluation. As a future work, this issue should be solved by increasing the size of the dataset to avoid dataset imbalance.

5 Discussion and Conclusion

In this study, we analyzed customers' sentiments on Twitter toward four Saudi Banks. A total of 50K raw tweets were scrapped using Twitter web scrapping tool. After data cleaning and preprocessing we ended up with 12k tweets. The dataset was manually annotated into three categories positive, negative, and neutral where we observed that among all classes the dominant tweets were pertained to negative sentiment. Additionally, as shown in Figure 3 we found that AlRajhi bank has the highest number of negative tweets followed by Saudi National Bank (SNB). Likewise, the positive and neutral tweets were the lower one.

We have utilized the annotated data to train three baseline classifiers namely Support vector machine (SVM), Random Forest (RF) and Logistic regression (LR) and fine-tuned one pretrained language model MarBERT. The baseline models were trained and tested on the same dataset which has been evaluated using 5-fold validation. Whereas MarBERT transformer model was trained and evaluated using the same evaluation method. The evaluation results show that the best accuracy result was achieved by logistic regression (LR) with 81.0 F1 score which outperforms the pre-trained model MarBERT that had achieved F1 score of 68%. Technically speaking, the accuracy result for all observed classifiers were very close to 82%. Other metrics have large differences between classifiers performance where the highest precision and recall results were 80% and 82% respectively which was achieved by two classifiers LR and SVM. Furthermore, we can conclude that overall SVM and LR models outperform the pre-trained model MarBERT in precision and recall as well as F1 score.

In the future, we plan to increase the dataset and try to prevent the overfitting problem using some techniques such as data augmentation as well as investigate the performance of deep learning models (e.g. BiLSTM and CNN) on the proposed dataset. We might also try to use ensemble models to improve the experiments results.

References

Almuqren, L., and Cristea, A. (2021). AraCust: A Saudi Telecom Tweets corpus for sentiment analysis. PeerJ Computer Science, 7, e510. https://doi.org/10.7717/peerj-cs.510

Al-Twairesh, N., Al-Khalifa, H., Al-Salman, A., and Al-Ohali, Y. (2017). AraSenTi-Tweet: A Corpus for Arabic Sentiment Analysis of Saudi Tweets. Procedia Computer Science, 117, 63–72. https://doi.org/10.1016/j.procs.2017.10.094

Azunre, P. (2021, July). Transfer Learning for Natural Language Processing. Manning Publications. https://www.manning.com/books/transfer-learning-for-natural-language-processing

Bessou, S., and Aberkane, R. (2019). Subjective Sentiment Analysis for Arabic Newswire Comments. Journal of Digital Information Management, 17(5), 289. https://doi.org/10.6025/jdim/2019/17/5/289-295

Di, L., Shaiban, M. S., and Hasanov, A. S. (2021). The power of investor sentiment in explaining bank stock performance: Listed conventional vs. Islamic banks. Pacific-Basin Finance Journal, 66, 101509. https://doi.org/10.1016/j.pacfin.2021.101509

Eksa Permana, M., Ramadhan, H., Budi, I., Budi Santoso, A., and Kresna Putra, P. (2020a). Sentiment Analysis and Topic Detection of Mobile Banking Application Review. 2020 Fifth International Conference on Informatics and Computing (ICIC), 1–6. https://doi.org/10.1109/ICIC50835.2020.9288616

Eksa Permana, M., Ramadhan, H., Budi, I., Budi Santoso, A., and Kresna Putra, P. (2020b). Sentiment Analysis and Topic Detection of Mobile Banking Application Review. 2020 Fifth International Conference on Informatics and Computing (ICIC), 1–6. https://doi.org/10.1109/ICIC50835.2020.9288616

Elamir, E. A. H., and Mousa, G. A. (2020). Sentiment Analysis of Banks' Annual Reports and Bank Features: LASSO Approach. 2020 International Conference on Decision Aid Sciences and Application (DASA), 42–48. https://doi.org/10.1109/DASA51403.2020.9317075

Gavval, R., Ravi, V., Harshal, K. R., Gangwar, A., and Ravi, K. (2019). CUDA-Self-Organizing feature map based visual sentiment analysis of bank customer complaints for Analytical CRM. ArXiv:1905.09598 [Cs]. http://arxiv.org/abs/1905.09598

Kazmaier, J., and Vuuren, J. van. (2020). Sentiment analysis of unstructured customer feedback for a retail bank. ORiON, 36(1), 35–71. https://doi.org/10.5784/36-1-668

Krishna, G. J., Ravi, V., Reddy, B. V., Zaheeruddin, M., Jaiswal, H., Teja, P. S. R., and Gavval, R. (2019). Sentiment Classification of Indian Banks' Customer Complaints. TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 429–434. https://doi.org/10.1109/TENCON.2019.8929703

Olson, D., and Delen, D. (2008). Advanced Data Mining Techniques. In Springer. USA. https://doi.org/10.1007/978-3-540-76917-0

Towards an Automatic Dialect Identification System for Algerian Dialects Using YouTube Videos

Khaled Lounnas
USTHB, Algeria
klounnas@usthb.dz

Mohamed Lichouri
CRSTDLA, Algeria
m.lichouri@crstdla.dz

Mourad Abbas
HCLA, Algeria
m_abbas04@yahoo.fr

Thissas Chahboub, Samir Salmi
USDB, Algeria
chahboubthissas@gmail.com
samirsalmi.eg@gmail.com

Abstract

Many academics are becoming more interested in Spoken Arabic Dialect Identification. Nonetheless, most under-resourced languages suffer from a lack of data, such as the common Algerian dialect, which provides an intriguing case study. As a result, the purpose of this research is to compare the performance of two techniques for the automated identification of Algerian dialects. The first is based on acoustic features whereas the second is based on spectral components extracted from audio sequences collected from YouTube. The experiments were carried out in two setups: raw data and noiseless data (applied noise filter) on 23 Algerian dialects using machine, deep, and transfer learning models while selecting three duration: 5s, 10s, and 20s. The CNN classifier performed the best, enabling us to generate an average F1score of 97.09% with raw data and 96.5% with noiseless data, independent of duration. However, the 20s duration result, which had an F1score of 98.09%, was the best duration that produced the best results for us.

1 Introduction

In all communication technologies, speech is the most natural mode for individuals to make direct contact. With the progress of technology, the scientific community has grown increasingly interested in the field of speech processing, seeking to explore and examine language and the process of voice generation. These tasks are very interesting, especially for low-resourced languages like Arabic and its dialects.

Algeria's dialect, with its richness and diversity, does not correspond to linguistic criteria since it differs from standard Arabic and is composed of a vocabulary with several sources. As a result, we have chosen a phonetic idea of the language rather than a linguistic one. Thus, automatic dialect recognition is the initial stage in performing numerous tasks in NLP (speech translation, opinion mining, etc.), and this study will be the first step in breaking down communication barriers between several Algerian areas.

The contribution of this paper is the development of an automatic spoken language identification system, employing a variety of machine and deep learning approaches to cover 23 classes of Algerian dialects acquired from YouTube videos.

This paper is organized as follows: we present an overview of both speech-based dialect identification and recognition of dialectal speech, and the related work in sections 2. In section 4, we present the system architecture. In section 3, we describe the corpus used to run different experiments. Sections 5 and 6 is devoted to experiments and results regarding dialect identification. The conclusion is presented in section 7.

2 Related Work

Many different sorts of studies have been conducted on spoken dialect identification; some have employed traditional approaches based on statistical classification (Korkmaz and Boyacı, 2022), while others have been enticed to apply deep learning techniques (Garain et al., 2021). However, relatively little study has been conducted on Arabic dialects (Biadsy et al., 2011; Bougrine and Abdelali, 2018; Lounnas et al., 2022).

In the case of Arabic spoken dialect identification, we refer to the research published in (Biadsy and Hirschberg, 2009), the authors used prosodic cues and demonstrated their efficacy across four main Arabic dialects, including Gulf, Iraqi, Lev-

antine, and Egyptian, to demonstrate how employing these descriptors to train the Gaussian Mixture Model (GMM) in conjunction with the Universal Background Model (UBM) may greatly enhance the identification of these dialects of 2-minute utterances. In keeping with their same area of study, the authors tackled the identification of the Arabic accent and dialect in (Biadsy et al., 2011). To do this, they employed phonetic segmentation supravector, which entails creating a kernel function that computes phonetic similarities in order to train the Support Vector Machine classifier. Their Equal Error Rate (EER) was 12.9 %.In (Ali et al., 2015), where researchers looked at several methods for dialect identification in Arabic broadcast speech based on phonetic and lexical characteristics received from a voice recognition system and bottleneck features created using the i-vector framework. They achieved 100% accuracy by employing a binary classifier to distinguish between Modern Standard Arabic (MSA) and dialectal Arabic. While they were able to distinguish between five Arabic dialects—Egyptian, Gulf, Levantine, North African, and MSA—with an accuracy of 59.2%. Authors in (Eldesouki et al., 2016) were concerned with recognizing spoken Arabic dialects from five regions, namely Egyptian, Gulf, Levantine, North-African (Maghrebi), and MSA. Despite the modest quantity of data employed, the researchers claimed that the Linear Support Vector Machine (LSVM) classifier trained with a feature vector incorporating textual features beat the other systems, achieving an accuracy of 51.36%. (Shon et al., 2020) supplied vast dialectal Arabic corpora encompassing 17 dialects to provide more resources for Arabic and its dialects. A total of 3000 hours of speech were provided for training a fine-grained Arabic dialects recognition system, which was divided into three groups based on time ($<$ 5 sec, 5 sec $\sim$ 20 sec, and $>$20 sec). Furthermore, several cutting-edge approaches were developed utilizing the aforementioned dataset, the results reveal that the longer the duration of the speech (in this case more than 20 seconds), the better its identification. Concerning the same issue, and to emphasize the use of the X-Vector approach in the identification of Arabic-spoken dialects, Hanani et al. (Hanani and Naser, 2020) created an X-Vector model utilizing a collection of relevant characteristics (acoustic, lexical, and phonetic) derived from VarDial 2018 and VarDial 2017 and shown that it outperforms existing state-of-the-art models, such as those based on i-vectors, Bottleneck features, and GMM-tokens.

However, for a vernacular Arabic dialect like the Algerian dialect, there are not many works have been done on it. We can cite the contribution of Bougrine et al. (Bougrine et al., 2016) in which she introduced the first Algerian spoken dialect corpora where six Algerian dialects have been modeled in (Bougrine et al., 2018) utilizing prosodic information, which is comprised of rhythm and intonation, and SVM based on the Universal Pearson VII Kernel function (PUK). The authors discovered that prosodic cue was appropriate even for brief utterances with a precision of over 69%. In (Terbeh et al., 2018), the authors suggested a statistical method based on phonetic modelling to determine the relevant Arabic dialect for each input acoustic signal by computing the necessary phonetic model, which was then compared to all referred Arabic dialect models using cosine similarity. (Lounnas et al., 2018) conducted a series of tests using various feature configurations to distinguish between Standard Arabic and one of the Berber dialects known as Kabyl [1]. They demonstrated that a combination of acoustic (Mel Frequency Cepstral Coefficients) and prosodic (melody and stress) characteristics are the best way to distinguish these dialects. A further extension of this work is the one developed in (Lounnas et al., 2019b), in which The difficulty of recognizing languages such as Persian, German, English, Arabic, and Kabyl has been handled by using the Voxforge voice corpus where different systems have been built to identify Persian, German, English, Arabic, and Kabyl dialects. Despite the small size of the data, the system produced an encouraging accuracy of 84.6%.

3 Comparison methodology

As described before, in this paper we focus on identifying Algerian dialects(Bougrine et al., 2016) using our own private dataset. To begin, we converted the videos into audio files. Following the conversion, the audios are pre-processed to reduce noise; this step is only performed for the acoustic approach. The audio files will be divided into 5s, 10s, and 20s segments. We use the segmented data to execute two distinct methods: the first entails extracting the dialects' acoustic parameters and assigning them to a single numerical vector (Lounnas et al., 2020) in order to classify them

[1] Kabyl is an Algerian Berber dialect

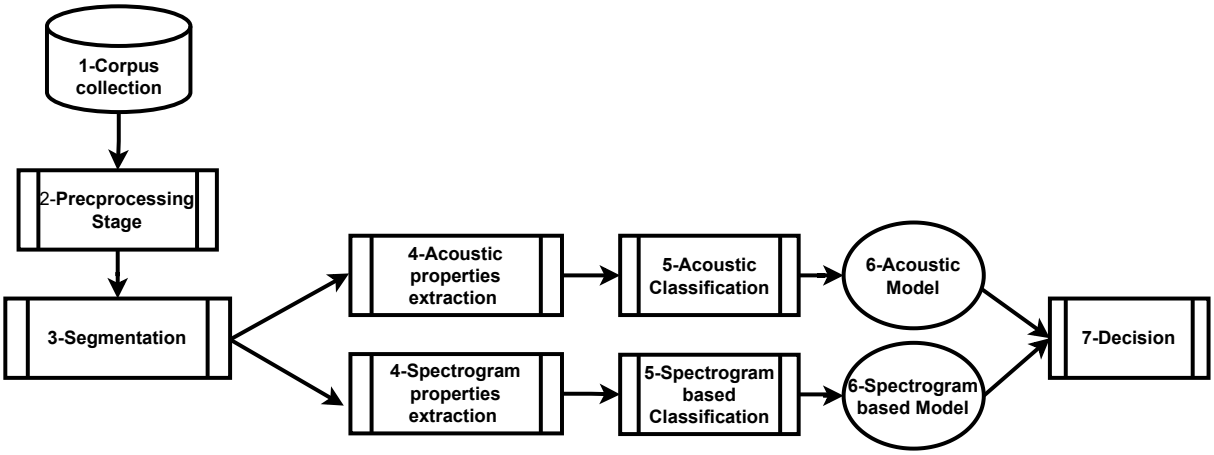


Figure 1: Architecture of our proposed spoken Algerian dialect identification system

using two models, a Support Vector Machine and a Convolutional Neural Network model. The second technique entails transforming the audios of each segment duration into spectrograms (Lounnas et al., 2022), and the resulting image will be pre-processed before being classified using a pre-trained Transfer Learning model VGG16, which has a high accuracy for image classification. Finally, we use assessment metrics to evaluate the trained model.

The schematic presented in Figure 1 depicts the overall design of our approach for the automated identification of Algerian dialects. The architecture is structured as a pipeline of multiple processes which are run sequentially: (1) collection of data; (2) preprocessing; (3) segmentation; (4) features extraction (acoustic vs spectrogram); (5) model training and finally (6) system evaluation. All these processes will be presented in the following sections.

4 Corpus presentation

We selected YouTube as a source for our work since it contains videos on a range of topics created by authors from all across the country. This allowed us to create an Algerian voice datasets with accentual and dialectal variations with around 30 hours of spoken audio. After selecting some YouTube channels that are related to food recipes, daily life, education, and monologue videos, we downloaded all the videos and saved them in MP4 format. The choice is explained by the nature of this video where they don't contain music in the background and with low noise levels. Because our corpus only has 23 cities, there are only 23 sub-dialect. We have various dialects from the center, west, and east that are presented in Table 1 with clips ranging in length from 2h7min to 2h42min for each dialect.

It should be noted that this is one of the fewer works, to our knowledge, that build an Algerian speech corpus for dialect identification, where the first corpus is named KALAM'DZ (Bougrine et al., 2016). In Table 2, which highlights the number of sub-dialects, overall duration, duration per sub-dialect, preprocessing processes, source of data, number of speakers per file, and use cases of each corpus, we compared our corpus to KALAM'DZ. Even though our corpus is smaller than KALAM'DZ's, there is one difference: our emphasis was on YouTube videos with extremely little background noise and no music.

Region	Departments
The North Centre	Algiers, Blida, Tipaza, AinDefla, Tizi Ouzou, Ténès
The North West	Tlemcen, SidiBelabbes, Oran, Maghnia
The North East	Jijel, Annaba, Guelma, Constantine
Central Highlands	M'sila , Laghouat, Bousaada
Eastern Highlands	Batna, Tebessa, Setif, Khenchela
Hoggar-Tassili	Tamanrasset

Table 1: Collected Algerian speech corpora department classification by geographical region (ONS, 2011)

Corpus	KALAM'DZ	Our Corpus
# sub-dialect	43	23
Duration H	104.4	around 47
DpSd H	13.05 (average)	around 2
Preproc	Non-speech segments removal; Speaker Diarization	Noise Reduction
Source	Algerian radio; Algerian TVs; YouTube; Podcast	YouTube
# speaker	Multi speaker	Monologue
Use cases	NLP	Dialect Identification

Table 2: A comparative study between our proposed corpus and the most useful and existing one KALAM'DZ DpSd: Duration per Sub dialect; Preproc: Preprocessing

4.1 Pre-processing the corpus

In Machine Learning, data pre-processing is an important step that helps improve data quality and promotes the extraction of relevant information from data. It is the process of preparing (cleaning and organizing) raw data so that it may be used to build and train Machine Learning and Deep Learning models. Simply stated, data preprocessing is a data mining approach that converts raw data into a comprehensible and legible format.

4.1.1 Audio preprocessing

We eliminate noise by utilizing Python's "NoiseReduce" library [2], which lowers noise in temporal data such as voice. It is based on a mechanism known as a "spectral gate," which is a type of Noise Gate. It works by computing a signal's spectrogram and predicting a noise threshold (or gate) for each frequency band of this signal/noise. This threshold is used to create a mask that filters out noise below the frequency variation threshold (Sainburg et al., 2020). As seen in Figure 2, the NoiseReduce library removes a considerable number of contaminants from our signal.

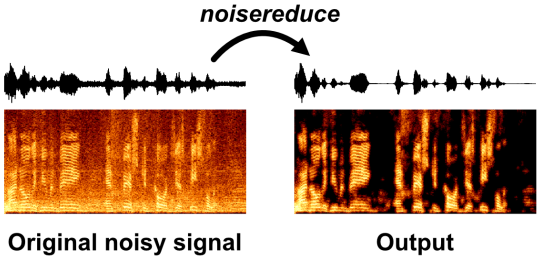


Figure 2: Removing noise from an audio file using the NoiseReduce library (Sainburg, 2019)

5 Experimental Setup

Our classification tasks begin with annotated audio data. There are several forms of audio classifications, but for the sake of our research, we are only interested in two: classification based on acoustic characteristics and classification based on spectrograms.

5.1 Acoustic-based approach

Acoustic information is commonly regarded as the initial level of processing in speech production. It is one of the simplest types of information that may be collected straight from raw speech during the speech parameterization process. Higher level speech information, such as phonotactic and word information, may also be retrieved from acoustic data. Linear Prediction, Mel Frequency Cepstral Coefficient (MFCC), Perceptual Linear Prediction (PLP), and Linear Prediction Cepstral Coefficient (LPCC) are the most often utilized parameterization approaches. We utilized a process based on Librosa (McFee et al., 2015), which incorporates spectral and rhythm properties. As in our previous work in (Lounnas et al., 2019a), we will use the same characteristics, with a total of 193 components:

1. MFCC coefficients (40)
2. Mel spectrogram (128) & Chroma Vector (12)
3. Spectral contrast (7) & Tonnetz(6)

[2] https://pypi.org/project/noisereduce/

These features served as the training data for both an SVM model as well as CNN.

5.2 Spectogram-based approach

We chose to implement the spectrogram method in order to compare the efficiency of the acoustical features extraction approach with that of the spectrogram feature extraction approach for audio clips. The goal is to learn how to classify audio and predict which category they belong to (dialect). We classify the audio from the image by looking at the spectrogram, which relates an intensity or a power to each frequency. The classification task in this situation seems as an image classification. This issue can be applied to a variety of practical applications, such as classifying music videos to determine the genre of music (Nirmal and Mohan, 2020; KM et al., 2021) or classifying short utterances by a group of speakers to identify the speaker based on voice (Liu et al., 2018).

6 Results and discussions

We used the **Sklearn** package with the default parameters to build our first SVM model. Concerning the CNN we will initialize our model as follows (Figure 3):

```
model.add(Conv1D(64, 3, activation='relu', input_shape = 193))
model.add(Conv1D(64, 3, activation='relu'))
model.add(MaxPooling1D(3))
model.add(Conv1D(128, 3, activation='relu'))
model.add(Conv1D(128, 3, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(23, activation='softmax'))
```

Figure 3: CNN model architecture

6.1 Acoustic features-based classification results

After collecting the video samples, we converted them to ".wav" format using the **fffmpeg** function with a sampling rate of 16k. For noise suppression, we utilized the **NoiseReduce** tools(Sainburg, 2019) at the preprocessing stage. In order To compare three different case studies, we divided our speech data into segments of three different duration (5s, 10s, and 20s), which would serve as the input data for three different models. We utilize the functions presented in **Librosa** library to extract the aforementioned characteristics. Each audio file of our corpus will go through this procedure, with the features being stacked vertically in one array vector and the labels in another. Both vectors will be stored in two files, "Features.npy" for the features and "Labels.npy" for the labels, at the end of the feature extraction procedure for later use. After the feature extraction stage, we divide our data into training data and test data, each comprising 80% and 20% of the corpus respectively. Finally, we were able to compare our 12 categorization models, and the following table presents the obtained accuracy and F1score (see Table 3).

For 5s segments, we see that both algorithms' (SVM and CNN) raw data results (95.55% and 96.89%) are actually superior to those obtained from preprocessed data (90.37% and 95.33%). For the 10-second segments, we see that the CNN model's predictions for preprocessed data (96.49%) are marginally better as compared to the raw data (96.31%). However, the SVM model with raw data surpasses the preprocessed data results by 5% (96.65% vs 91.71%). The raw data results outperform the preprocessed data for the 20s segments for both algorithms. We find a minor decline in performance for the preprocessed data, which is related to the fact that it is not as good as the raw data, which is due to noise removal loss.

6.2 Spectogram features-based classification results

The corpus utilized is identical to the one mentioned in the preceding section. We next turn the segmented audio into their spectrograms template. Given that we were working with spectrograms representation, we couldn't utilize SVMs as it is; otherwise, we'd have to add a feature vector extraction phase, so we opted to work with a pre-trained model called Transfer Learning (TL). As there are many TL models, we selected one of the smaller models in term of parameters numbers which is the visual geometric group (VGG16).

According to the results described in Table 4, identifying dialects using spectrograms achieved its best performance with 20s duration data with a macro F1score of 88%, while the scores for 10s and 5s audios are 84% and 57%, respectively. These results are appropriate since the 20s audios include more information than the 5s and 10s. We noted that the duration of the audio file may affect the number of epochs needed for the CNN, where the 5s, 10s, and 20s have required epochs 3, 5, and 20, respectively. To summarize, the pre-trained VGG-16 model didn't achieve the results that were attended, if we take into account that it is well-known for its great accuracy in image classification. We

	Model\Data	*Raw data*			*Preprocessed data*		
		5s	**10s**	**20s**	**5s**	**10s**	**20s**
Accuracy	**SVM**	95.52	96.63	95.77	90.36	91.7	91.94
	CNN	96.84	96.27	98.08	95.29	96.48	97.67
F1score	**SVM**	**95.55**	**96.65**	**95.79**	90.37	91.71	91.92
	CNN	**96.89**	96.31	**98.09**	95.33	**96.49**	97.69

Table 3: Algerian Dialect Identification Based Acoustic features: Reported Result Before and After Processing (noise reduction). The best F1score obtained are in bold.

	5s	**10s**	**20s**
F1score	54	84	88
Epoch	3	5	20

Table 4: Results obtained by the classification of spectogram images using VGG16 pretrained model. The best epoch for each duration is reported.

think that the problem resides in the architecture of the network that we used to retrain the VGG-16 (2-layer network).

By comparing the performance of both the acoustic and spectrogram approach based on the obtained findings, we noted that the acoustic-based approach performs way better than the spectrogram-based approach (with an increase of about 10%). This suggests that auditory features are more trustworthy descriptors for distinguishing various dialects, reducing confusion, and producing better outcomes. The acquired findings are quite good when compared to what is available in the literature; our addition to this work is that we worked with 23 Algerian dialects.

7 Conclusion

This research focuses on the creation of an automatic system for recognizing Algerian dialects. To attain our aim, we employed two approaches. The first is based on acoustic features derived from audio, while the second is based on spectrogram features. The two approaches have been evaluated using SVM and CNN for the first one whereas a transfer learning technics (VGG-16) was applied for the second approach, respectively.

These models were evaluated using audio durations of 5s, 10s, and 20s. The results for the 20s duration data using CNN were extremely good, with an accuracy of 98.09% for the raw data. However, it should be highlighted that while employing a spectrogram to train a VGG deep learning model, our proposal performed best when the size of the audio voice was significant (the 20s). The previous experience with the 23 Algerian dialects stresses the relevance of acoustic parameters and the use of spectrograms in differentiating Algerian dialects. Future research might try to add new dialects and cities to our current corpus in order to eventually encompass all Algerian dialects. In addition, we will investigate deep learning methods to improve the modeling of Algerian dialects. Finally, we will look for the optimal duration that will allow our system to generalize more effectively.

References

Ahmed Ali, Najim Dehak, Patrick Cardinal, Sameer Khurana, Sree Harsha Yella, James Glass, Peter Bell, and Steve Renals. 2015. Automatic dialect detection in arabic broadcast speech. *arXiv preprint arXiv:1509.06928*.

Fadi Biadsy and Julia Bell Hirschberg. 2009. Using prosody and phonotactics in arabic dialect identification. In *Academic Commons, https://doi.org/10.7916/D8HM5HRV*.

Fadi Biadsy, Julia Bell Hirschberg, and Daniel PW Ellis. 2011. Dialect and accent recognition using phonetic-segmentation supervectors.

Hadda Cherroun Soumia Bougrine and Ahmed Abdelali. 2018. Spoken arabic algerian dialect identification. In *2018 2nd International Conference on Natural Language and Speech Processing (ICNLSP)*, pages 1–6. IEEE.

Soumia Bougrine, Hadda Cherroun, and Djelloul Ziadi. 2018. Prosody-based spoken algerian arabic dialect identification. *Procedia Computer Science*, 128:9–17.

Soumia Bougrine, Hadda Cherroun, Djelloul Ziadi, Abdallah Lakhdari, and Aicha Chorana. 2016. Toward a rich arabic speech parallel corpus for algerian subdialects. In *The 2nd Workshop on Arabic Corpora and Processing Tools*, pages 2–10.

Mohamed Eldesouki, Fahim Dalvi, Hassan Sajjad, and Kareem Darwish. 2016. Qcri@ dsl 2016: Spoken arabic dialect identification using textual features. In

Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3), pages 221–226.

Avishek Garain, Pawan Kumar Singh, and Ram Sarkar. 2021. Fuzzygcp: A deep learning architecture for automatic spoken language identification from speech signals. *Expert Systems with Applications*, 168:114416.

Abualsoud Hanani and Rabee Naser. 2020. Spoken arabic dialect recognition using x-vectors. *Natural Language Engineering*, 26:691 – 700.

Athulya KM et al. 2021. Deep learning based music genre classification using spectrogram.

Yunus Korkmaz and Aytuğ Boyacı. 2022. A comprehensive turkish accent/dialect recognition system using acoustic perceptual formants. *Applied Acoustics*, 193:108761.

Zheli Liu, Zhendong Wu, Tong Li, Jin Li, and Chao Shen. 2018. Gmm and cnn hybrid method for short utterance speaker recognition. *IEEE Transactions on Industrial informatics*, 14(7):3244–3252.

Khaled Lounnas, Mourad Abbas, and Mohamed Lichouri. 2019a. Building a speech corpus based on arabic podcasts for language and dialect identification. In *Proceedings of the 3rd International Conference on Natural Language and Speech Processing*, pages 54–58.

Khaled Lounnas, Mourad Abbas, Mohamed Lichouri, Mohamed Hamidi, Hassan Satori, and Hocine Teffahi. 2022. Enhancement of spoken digits recognition for under-resourced languages: case of algerian and moroccan dialects. *International Journal of Speech Technology*, pages 1–13.

Khaled Lounnas, Mourad Abbas, Hocine Teffahi, and Mohamed Lichouri. 2019b. A language identification system based on voxforge speech corpus. In *International Conference on Advanced Machine Learning Technologies and Applications*, pages 529–534. Springer.

Khaled Lounnas, Lyes Demri, Leila Falek, and Hocine Teffahi. 2018. Automatic language identification for berber and arabic languages using prosodic features. In *2018 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM)*, pages 1–4. IEEE.

Khaled Lounnas, Hassan Satori, Mohamed Hamidi, Hocine Teffahi, Mourad Abbas, and Mohamed Lichouri. 2020. Cliasr: a combined automatic speech recognition and language identification system. In *2020 1st international conference on innovative research in applied science, engineering and Technology (IRASET)*, pages 1–5. IEEE.

Brian McFee, Colin Raffel, Dawen Liang, Daniel P Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25. Citeseer.

MR Nirmal and Shajee Mohan. 2020. Music genre classification using spectrograms. In *2020 International Conference on Power, Instrumentation, Control and Computing (PICC)*, pages 1–5. IEEE.

ONS. 2011. Recensement général de la population et de l'habitat – 2008 – (résultats issus de l'exploitation exhaustive).

Tim Sainburg. 2019. timsainb/noisereduce: v1.0.

Tim Sainburg, Marvin Thielk, and Timothy Q Gentner. 2020. Finding, visualizing, and quantifying latent structure across diverse animal vocal repertoires. *PLoS computational biology*, 16(10):e1008228.

Suwon Shon, Ahmed Ali, Younes Samih, Hamdy Mubarak, and James Glass. 2020. Adi17: A fine-grained arabic dialect identification dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8244–8248 . IEEE.

Naim Terbeh, Mohsen Maraoui, and Mounir Zrigui. 2018. Arabic dialect identification based on probabilistic-phonetic modeling. *Computación y Sistemas*, 22(3):863–870 .

Constructing the Corpus of Chinese Textual 'Run-on' Sentences (CCTRS): Discourse Corpus Benchmark with Multi-layer Annotations

Kun Sun
University of Tübingen, Germany
kun.sun@uni-tuebingen.de

Rong Wang
University of Stuttgart, Germany
rong4ivy@163.com

Abstract

Combining the annotation strengths of PDTB and RST, this study constructs a specialized Chinese discourse corpus on "run-on" sentences. "Run-on" sentences are a typical and prevalent form of discourse/text in Chinese. Despite their widespread use in Chinese, previous studies have only explored "run-on" sentences by using small-scale examples. In order to carry out computational tasks in realistic context and increase diversity of discourse corpora resources, we establish this discourse corpus. The present study selects 500 "run-on"sentences and annotates them on the levels of discourse, syntax and semantics. We mainly adopt an integrated annotation pipeline combining with RST and PDTB to process these sentences. After that, three state-of-the-art discourse parsers are employed to test the feasibility of this corpus, and the result shows that this corpus performs stably and can be used as a benchmark for evaluating discourse parsing.

1 Introduction

Discourse corpora annotated with discourse relations have become important in many down-stream NLP tasks including machine translation (Guzmán et al., 2014), machine reading comprehension (He et al., 2017) and automatic summarization (El-Kassas et al., 2021). Several discourse corpora have been proposed in previous work, grounded with various discourse theories. Currently there have been two influential discourse annotation systems: PDTB (Penn Discourse Treebank) and RST (rhetorical structure theory) (Mann and Thompson, 1988; Carlson et al., 2003; Webber, 2004; Webber et al., 2019). The two systems have their own strengths. However, few discourse corpora could have annotated using the strengths from the two systems. The two annotation systems have been adopted to annotate discourse structure in a few languages. However, for example, few Chinese corpora were both annotated for discourse properties and publicly available (Zhou and Xue, 2015; Jiang et al., 2018). The available annotated texts are primarily newspaper articles. The other problem is that these annotated Chinese discourse corpora seldom annotated other relevant information considering the characteristics of the Chinese language.

The Chinese language is known to be a discourse-oriented language (Tsao, 1979; Chu, 1998; Li, 2005). It is characterized by a very common but special linguistic phenomenon that is called "run-on" sentences. Native Chinese linguists usually refer to this type of sentence as *liu shui ju*, a "flowing-water sentence" (流水句), in which the metaphor "liu shui" (flowing water) vividly describes the physical feature and the logical relationships between the segments of such a sentence. Linguists working on the Chinese language often boast about the special characteristics of "run-on" sentence as such sentences may be seen as grammatically unacceptable in English but they are nonetheless widespread in Chinese (Sheng, 2016; Wang and Zhao, 2017). However, the term "run-on" does not describe the characteristics of these Chinese sentences in a precise way. Instead, this term simply helps those unfamiliar with the Chinese language to understand what they are. A "run-on" English sentence is like this: "*My cat meowed angrily, I knew she wanted food, I hurried to go to shop and make her food.*" By contrast, a Chinese "run-on" sentence would read as follows "*My cat meowed angrily, I hurried to go to shop and make her food.*" In other words, when the middle clause in the English sentence is missing, it feels that there is a semantic leap and it resembles a "run-on" sentence in Chinese.

Linguists working on Chinese have been aware of this phenomenon for a long time (Lu and Zhu, 1979; Hu and Jingsong, 1989; Chao, 1968; Shen, 2012). According to previous research, this means the general characteristics of "run-on" sentences can be summarized phonetically, syntactically, and

semantically. By contrast, English, Japanese, and Korean do not have "run-on" sentences. The compound sentence structures are closed in these languages and usually consist of multiple clauses. The clauses are linked through logical relationships such as parallelism, causation, and concession. That means there are clear boundaries between single, complex, or compound sentences in these languages. This is not the case with Chinese run-on sentences. "Run-on" sentences seem to be "sentences". However, a "run-on" sentence is actually a discourse although it is composed of several segments enclosed by a full stop. The reason for this is that there is no clear boundary between the sentence and the text in many cases.

Further quantitative/computational research on "run-on" sentences and realistic Chinese discourse requires annotating a corpus and thus obtaining data. This study targets to construct the corpus of Chines textual "run-on" sentences (CCTRS). Clearly, discourse relations are the core characteristics of textual "run-on" sentences. As discussed at the outset, borrowing the annotation styles from the two mature discourse corpora (PDTB, RST), we annotated the discourse relations for "run-on" sentences. Although some discourse parallel corpora were created using PDTB and RST piplelines separately (Potsdam Commentary Corpus, Stede and Neumann, 2014; GUM corpora, Zeldes, 2017), no discourse corpora have merged the two pipelines into one integrated system to annotate discourse relations previously. The CCTRS is the first to do this.

This corpus (CCTRS) is to provide a benchmark of realistic datasets in Chinese computational discourse analysis. Compared with the past discourse corpora, the CCTRS accommodates an integrated method to annotate discourse relations and makes multilayer annotations regarding other semantic and discourse information. We believe that these annotation data are to promote further development on discourse relation recognition and discourse-level NLP tasks in realistic context. Further, the CCTRS can increase diversity of discourse corpora resources. The data from the CCTRS can help investigate linguistic problems quantitatively and promote the improvement of algorithms for zero anaphora resolution from the perspective of discourse relations.

2 Related Work

The question of how discourse units are effectively incorporated into a unified meaningful text has been addressed from a variety of perspectives, such as Hobbs's theory of coherence relations (Hobbs, 1979), the rhetorical structure theory (Mann and Thompson, 1988), the centering theory (Grosz et al., 1995), the discourse representation theory developed by (Asher et al., 2003), and the framework of lexicalized tree-adjoining grammar (L-TAG, Webber (2004)). Annotations on aspects of discourse structure have been made in text corpora on the basis of these theories.

L-TAG theory holds that discourse relations can be lexicalized, implying that two clauses linked by a connective contribute to two distinct arguments. Adopting this lexically-grounded predicate-argument approach, the Penn discourse treebank (PDTB3.0, Webber et al., 2019) provides annotations of discourse structures for English. However, Chinese discourse uses very few conjunctions and connectives (75% discourse connectives are implicit). Zhou and Xue (2015) followed the PDTB guidelines in establishing Chinese discourse treebank (CDTB). The other two Chinese discourse corpora were established following the PDTB styles (Zhou et al., 2014; Long et al., 2020). RST(rhetorical structure theory), another influential discourse theory, assumes there is a hierarchy of discourse segments that collectively span a full text. The RST discourse treebank (RST-DT, Carlson et al., 2003) has been adopted to annotate discourse in a variety of languages. Nevertheless, in attempting to apply RST to the annotation of Chinese discourse, we find that there is in many cases no way of distinguishing between a nucleus and its satellite.

We illustrate the characteristics of a "run-on" sentence in Chinese using an example, which consists of a sequence of clauses, with its English translation (see Example 1 in the **Appendix**). Example 1 helps us understand why an integrated annotation system combining PDTB and RST was taken in the CCTRS. The subsection 3.2.1 will give a detailed account of why and how the two systems were taken to integrate into an annotation pipeline. This Chinese example is a typical "run-on" sentence. First, the sentence has no connectives that clarify the temporal and logical relations between the clauses. With the semantic relationships between clauses left implicit, interpreting the sentence may

sometimes require a considerable creative effort on the part of the reader. For example, after introducing the cry from a child, the sentence directly moves ahead to the clouds in the sky without mentioning the missing link of "*I felt bored and raised my head to find*", leaving a gap in the logical progression from cause to effect. The semantic leap between (3) and (4) leaves the reader/listener much more leeway in filling the gap. By contrast, an English translation usually provides the missing information for the semantic leap between (3) and (4), such as "*Strangely enough, when raising my head, I found that...*". Nevertheless, in English, we can find a similar phenomenon in discourse. It is possible that neither a discourse relation nor entity-based coherence can be inferred between the adjacent sentences. The phenomenon in English is actually the same as the semantic leap in Chinese "run-on" sentences. In the Penn Discourse Treebank (PDTB, Webber et al., 2019), such semantic leap in English discourse is annotated as "`NoRel`"(no relationship), which occurs with a very low frequency (0.67%).

Considering the uniqueness of "run-on" sentences, we also annotated the other types of information at grammatical, semantic and discourse levels. For example, topic chain plays a key role in discourse coherence. Semantic information, such as animacy also is very helpful in recognizing run-on sentences. It is the first time to annotate them in Chinese corpora. In the following, we will introduce them separately.

3 Annotation Scheme

3.1 Text and "run-on" sentence selection

The criterion of "run-on" sentences chosen for the corpus is easy to define and annotate while taking into account various research contributions. Take the following as an example, which is a direct translation from Chinese (in order to save the paper length, and the original example is in the Appendix).

[Example 2] *The river was full of people (1), four long vermilion boats were sliding in the pool(2), the water of the dragon boat had just risen(3), the water in the river was all bean green(4), the weather was so bright(5), the drums were sounding(6), [`semantic leap` (≈ `NoRel` in the PDTB)] Cuicui pursed her lips without saying a word(7), her heart was full of unspeakable joy(8).*

Clearly a semantic leap occurs from the sixth clause to the seventh clause, i.e., from the scene of dragon boat racing on the river to Cuicui directly and we do not know what relationship defines the discourse semantic relationship between (6) to (7). Although the definitions of "run-on" sentences are different, they all acknowledge the existence of semantic leaps between segments, which means this is an appropriate standard for selecting "run-on" sentences. *Semantic leap* is quite similar to `NoRel` in PDTB. We investigate these sentences both formally and semantically. Such a procedure is conducive to annotation and obtaining data. The second criterion is that all segments must be enclosed by a sentential-final period. In Example 2, clause (8) ends with a period. Although semantic leaps often occur between different sentences, these are not considered in this study. This is because we only interested in how semantic leaps occur within a block of clauses that native Chinese speakers/readers perceive as having a complete meaning. In short, we followed the two criteria to select "run-on" sentences. Note that the two criteria were selected from the characteristics described by previous related studies, and not from our own.

Further, in many cases, it is not easy to distinguish which clause is head or which clause is subordinate given that we attempt to establish head/subordinate between two clauses. For example, among the former six clauses in Example 2, it is hard to confirm which clause is a head. This indicates that the distinction between nucleus and satellite in the RST may not be applied in many cases, particularly in fiction genre.

As is well-known, multiple clauses can be joined by using commas without conjunctions in Chinese texts, with the period occurring at the end of the block of clauses and indicating the completeness of the meaning or idea therein rather than the completeness of a sentential structure (Lu and Zhu, 1979:322; Huang and Xiao, 2016; Xue and Yang, 2011, Sun and Lu, 2022). In this study, the selection of "run-on" sentences is limited to the fiction genre. 500 "run-on" sentences (equal to 500 texts) were carefully selected from ten well-known contemporary Chinese fiction (with 2.6 million Chinese characters in total).

3.2 Annotations at different levels

Combining the theoretical study of "run-on" sentences and the observation of examples, we venture the hypothesis that the factors contributed to the

characteristics of run-on sentences involve verb valence, clause structure, discourse semantics, topic chain, and other kinds of semantic information surrounding the clause with a semantic leap. As mentioned above, "run-on" sentences are actually treated as text/discourse. The discourse structure should be highlighted. The semantic leap is one of the most important traits of "run-on" sentences, which means we need to pay particular attention to the semantic status of the clause containing the semantic leap. We took these aforementioned factors into account and annotated them. These annotations can be classified into three separate types that represent the core features of "run-on" sentences, namely, discourse, grammatical and semantic. The following details these annotations in this corpus.

Sense	Class
Temporal	Succession (Su) Precedence (Pr) Simultaneous (Si)
Contingency	Cause-effect (Ce) Conditional (Co) Purpose (Pu)
Comparison	Contrast (Cn) Concession (Cc) Conjunction (Cj) Leap (Le)
Expansion	Continuation (Cu) Progression (Pg) List (Lt)

Table 1: The hierarchy of discourse structure for annotation tags in the CCTRS. There are only two-class hierarchy tags: sense and class.

3.2.1 Discourse relations

The annotations of discourse structure and discourse relations for the current corpus combine two of the most successful annotation systems (PDTB and RST-DT) and they take into account the characteristics of Chinese discourse. The following provides a detailed account of them respectively.

The English PDTB annotation system is a shallow discourse annotation, which is reasonable, neat, and easy to operate. RST semantic labels are more numerous and repetitive, while PDTB semantic labels are hierarchical and consistent. However, the PDTB cannot capture the global coherence. The CCTRS uses the semantic tagging of the PDTB and CDTB. However, unlike PDTB and CDTB, we did not explicitly annotate the discourse connectors (e.g., "because of", "despite"). There are two reasons for this: first, the number of explicit discourse connectors in "run-on" sentences is quite small. Second, the discourse connectors are supposed not to influence discourse relations becuase of few discourse connectives used in Chinese discourse (75.7% discourse connectives are implicit in Chinese).

Specifically, given the unbalanced data on PDTB relations, the samples are sparse. For instance, in the PDTB, *temporal* has three classes, but *contingency* includes eight classes. The CCTRS annotation system therefore has at most four specific labels under each sense. The high-frequency semantic relations in previous Chinese discourse are selected as tags. Due to the semantic peculiarities between the segments of the "run-on" sentences, two new semantic tags such as *leap*, and *continuation*. were taken in the annotation system. This makes it easier to compare this with the CDTB and with the PDTB corpora of other languages.

The majority of specific semantic labels were adopted from the PDTB rather than RST-DT. We also added two tags which do not occur in PTDB or CDTB. Table 1 shows the tags for discourse relations in our corpus. For example, here *Le* as tag is used to represent the discourse semantic leap relationship. Once *Le* appears, this means there is a semantic leap, so we can posit that a flow sentence is formed. The semantic leap (*Leap*) in "run-on" sentence is similar to `NoRel`(no relationship) in the PDTB. When talking about semantic leaps, we mean that no proper discourse relation can be employed to describe the semantic relation between the two segments. An English example from the PDTB illustrates the similarity between these phenomena between English and Chinese (See Example 3 in the Appendix). In Example 3, there is a semantic leap in the place of `NoRel`. NoRel indicates that a semantic leap occurs between two discourse units. According to the PDTB, these are cases where no discourse relation or entity-based coherence relation can be inferred between adjacent sentences. As a matter of fact, `NoRel` indicates that a semantic leap occurs between two discourse units. However, there are some differences between a semantic leap in Chinese and a `NoRel` in English. The first difference is that a semantic leap occurs within a "sentence", at least in a unit enclosed by a full stop in Chinese. However, `NoRel` in English seldom occurs within a sentence. According to PDTB3.0 manual, there is not any case where `NoRel` occurs within a sentence. The other difference is that the frequency of `NoRel` use

in English discourse is quite low. There are 254 NoRel cases in the PDTB, out of a total of annotated 40600 discourse relations. This suggests that NoRel cases only account for quite a small proportion of English discourse (about 0.63%). Although there is no direct statistical evidence concerning semantic leaps in Chinese, we believe that semantic leaps in discourse occur much more frequently than English according to our observation and the relevant studies. This claim is based on our observations and on a number of studies concerning Chinese "run-on" sentences.

The other new tag is *continuation*. It describes successive actions, or several events in succession. The tag of *continuation* was annotated when explicit time adverbials do not occur. Otherwise, the relation with explicit time adverbial was taken as *succession*. The relation of *continuation* is similar to *progression* in which one discourse unit represents a progression from the other, in extent, intensity, scale, etc. In contrast, there is no progression in extent, intensity etc. for *continuation*. In Chinese, they are defined by two different terms respectively, 承接 and 递进 .

We used RST constituent tree structure to annotate discourse structure for "run-on" sentences such that we can obtain the global coherence information. However, in view of the characteristics of Chinese discourse structure, we did not use the concepts of nucleus and satellite in the RST style annotation, because the relationship between nucleus and satellite is not very significant in some cases of fiction genre. Hence we did not annotate which is the nucleus and which is the satellite. For example, in Example 2, clauses 1-6 form a node ("1-6") and clauses 7-8 forms the other node ("7-8"). The first node ("1-6") has a "conjunction" relation with the other node ("7-8"). However, we almost cannot distinguish which node is the nucleus. More details can be seen in subsection 4.2.

According to Demberg et al. (2019), 76% of PDTB relations can be mapped with RST ones in the same English texts (53% directly mapped). This provides evidence that RST could be closely correlated with PDTB, that is, RST and PDTB could be merged together to make annotations for Chinese discourse. In short, our corpus used RST and PDTB strengths to annotate discourse structure and relations. We used RST tree to represent discourse structure but abandoned the concept of neclearity and satellite. We used two-class PDTB tags to annotate discourse relations but did not depend on discourse connectives.

3.2.2 Topic chain

When a clause does not have an argument before the predicate verb, it is recognized as co-referential zero amphora. The topic chain is considered as having a topic followed by several comment clauses, but usually the topic in the comment clauses is invisible, that is, is a *co-referential zero anaphora*. Within a topic chain, the topic controls and manages its comment clauses and the comment clauses are linked coherently, being dependent on some mechanism (Sun, 2019). Topic chain can help improve the performance of zero anaphora resolution in Chinese discourse (Kong et al., 2019). According to our observation, we feel that topic chain is closely related with discourse relations. Our follow-up study shows that topic chain is a good predictor for the occurrence of semantic leap in discourse. Here we annotated the number of clauses between the implicit topic and zero anaphor. This is called the *topic distance*, and it is mutually affected with discourse relations.

3.2.3 Grammatical status

The grammatical structure and the number of the *valence* of predicate verb in each clause was annotated. In English, a clause is the combination of a subject and a verb. There are two types of English clauses. Independent clauses consist of a subject and verb that make up a complete thought and they make sense on their own. In a dependent clause, the subject and verb don't make up a complete thought. Dependent clauses always need to be attached to an independent clause as they are too weak to stand alone. However, Chinese is very likely to use some VPs or NPs independently in discourse and these phrases can work independently as finite clauses in some cases. These independent VPs or NPs in Chinese are similar to English dependent clauses. The difference is that without the use of any grammatical device, these VPs and NPs can work independently. We use the following tags to annotate the grammatical status of each clause: *SVO (subject + predicative verb + object), SV (subject + predicative verb), SVC (subject + predicative verb + complement), VP (verb phrase), NP (noun phrase), and AP (adjective phrase).*

The *valence* indicates the number of arguments that are associated with a particular verb in a sentence. Most verbs are at least mono-valence. This

means they have one argument, which is the subject of the sentence that performs the *action* stated by the verb. There are also divalent verbs, which require both a subject and a direct object upon which the action is performed, and trivalent verbs that also need an indirect object that is part of the action. Valence is related to transitivity of verbs, although these are not identical concepts as the transitivity is based purely on objects and not the subject (Gao et al., 2014). We used the numeric value to record the valence information, which is based on the number of arguments of the predicate verb in a clause.

3.2.4 Semantic information

The animacy information and action information were annotated. The term *animacy* was explicated by Comrie (1989), who listed the hierarchical sequence for nominative entities according to the degree of animacy. Comrie (1989) outlined how the grammatical or semantic characteristics of nouns are dependent on how sentient or 'alive' the referent of a noun is. Animacy can have different effects on the grammar of a language, such as the choice of pronoun (what/who), case endings, word order, or the form a verb takes when associated with a noun. These constitute the animacy degree. However, using the traits of a semantic leap between two clauses as our criteria, we use only three animate features in making the annotations: *human*, *nonhuman*, and *inanimate*.

4 Annotation Methods

4.1 Annotation procedures

We then divided a "run-on" sentence into multiple segments. A segment is discourse unit, that is, roughly a unit of discourse that makes sense in pairs and individually. Generally the first thing to look at is the unit separated by commas. If it contains a verb, then this unit can be treated as a sentence or clause segment. If the segment separated by commas does not contain a verb, it may be a noun phrase that must be based on the situation, that is, the noun phrase is part of the content of the small sentence behind or in front. If this is the case, it cannot be divided independently. Sentence segment here is similar to discourse units or elementrary discourse units (EDU) in RST or PDTB. Generally, segment division is not easy, so two of the three annotators first divide the segments.

In fact, there are three levels of sentence annotation: grammatical, semantic, and discourse. Among them, a discourse relation between segments is used to mark not only the semantic relationship between two adjacent segments, but also the relationship between cross-segments (i.e., constituency structure), as described above. Grammatical annotation focuses on the syntactic form of each segment, whether it is a subject-predicate structure, a noun phrase, and also on the valence of the verb from the number of elements in the argument. The semantic annotation mainly focuses on the animacy of the subject noun of the segment before and after the semantic leap relation and whether or not the verb is dynamic or static. It is explained below using sample examples.

The three annotators are native Chinese speakers with linguistics background (two are MA students majoring in linguistics and the third one has obtained PhD degree in linguistics). The three annotators received training from the authors and achieved over 80% agreement on six pilot passages. They then independently annotated all sentences and met to resolve all discrepancies. We used standard corpus annotation methods to check the reliability of our annotations, which will be presented in the following subsection.

4.2 Sample example

After a "run-on" sentence was segmented (into EDU), we annotated each segment (EDU). The following example illustrates how to annotate a "run-on" sentence. All 500 "run-on" sentences (texts) were annotated in the same manner. An example (Example 4) is used to illustrate annotations (here is the English translation, and the original example is in the Appendix).

[Example 4] *At this time an airplane flew over(1), and a white band trailed behind the airplane (2), and lasted for a long time(3), and the sky was cut open (4), or the sky cracked (5), and leaked(6), and the fish disappeared (7).*

This sentence in Example 4 is divided into seven sentence segments (EDU). The grammatical structure is used to mark the structure of each segment. For example, the first segment, "the plane flew over", is a "subject-predicate" structure, which is represented by "SV"; the third segment, "long-lasting", is a verb phrase, which is represented by "VP". The semantic sequence, which semantically relates two segments or groups of segments to each other, is indicated by "1*2" for the first segment

and the second segment and "1-3*4" for the first to third segment and the fourth segment. All these segments can construe an RST tree structure where head or nucleus for each node is not distinguished, shown in Figure 1 in the Appendix. The semantic relation label adopted from the PDTB was introduced above. Here we need to introduce a new concept, that is, the *relation distance* is the linear distance between the segments forming a discourse semantic relationship (Sun and Xiong, 2019), e.g., "1*2" is "2-1=1", and "1-3*4" is "4-1=3". When a segment group is present, it is indicated by "1-3*4". When a number of sentence segments are linked by "-", such as "1-3", it indicates that these sentence segments are formed into one integrated unit, which is similar to RST structure.

The verb valence is the number of valences of the verb in the segment, for example, the verb "fly over" in the first segment has only one subject, so the number of valences is "1". The verb "not loose" in the third segment has a valence of 0. The *topic distance* is the number of segments between the occurrence of a subject that is omitted but that may occur in subsequent segments. Animacy and action are specifically true of the state of the subject and verb in the two segments where a semantic leap occurs. Animacy refers to whether the subject is a living being or a human being, "ina" stands for "inanimate", "nonhum" for "nonhuman", and "hum" for "human". Only three most distinctive features on animacy were chosen for simplicity and convenience. The action refers to the state in which the verb appears, whether it is dynamic or stative (Bach, 1986; Mcintosh, 1977), "dyn" is dynamic, and "sta" is stative. The main annotation information in Example 4 is shown in Table 2. [1]

4.3 Annotation reliability

A consistency assessment can be used to measure the objectivity of the corpus annotations. The assessment of the consistency of annotations provides a more objective picture of the quality of the annotation. The CDTB and the PDTB can assess the consistency of certain metrics such as discourse relations. To eliminate inconsistent annotations, we also used Kappa values (Siegel and Castellan, 1981) to assess the consistency of annotations evaluated using protocol rates. The final assessment of the degree of agreement between three annotators regarding the 500 sentences is shown in Table 3. The agreement ratio of the corpus is greater than 80% and the kappa value is greater than 0.6 for discourse structure, kernel, and relations. Krippendorff (1980) states that a k value greater than 0.6 for annotated data indicates good quality annotation. Table 3 shows that the annotations of the three annotators were very consistent as was the kappa perspective. This suffices to prove the reliability of our annotations.

5 Corpus Statistics

5.1 Frequency distribution

We look at simple statistics, mainly the frequency distribution of the various types of annotations in the CCTRS. With 500 sentences (texts) from the fiction genre, the CCTRS annotated 2286 discourse relations and 650 topic chains. Moreover, 954 clauses and 990 clauses were annotated by animacy and action information respectively. 2281 verbs were annotated by valency information. Figure 1 in the Appendix shows the distribution of frequencies for different discourse relations in the three discourse corpora. "Leap" occupies the largest proportion in the CCTRS, by contrast, "conjunction" takes the largest share in both the CDTB or the PDTB.We compare the distribution of frequencies among the three corpora. As shown in Figures 2 & 3 in the Appendix, the distribution of the three groups of data is basically similar. Generally speaking, such distribution of frequencies abides by the power law (Kello et al., 2010; Sun and Zhang, 2018).

5.2 Corpora comparisons

So far there have been four discourse corpora in Chinese, annotated in terms of PDTB and RST respectively. The four existing Chinese discourse corpora were annotated just with discourse relations. Compared with the four discourse corpora, the CCTRS annotations contain discourse relation, semantic information, grammatical structure and topic chain. This is the only multilayer annotation corpus. The differences among these corpora are seen in Table 4. A great deal of semantic and discourse information cannot be implemented through automatic annotation. In particular, there are very few corpus resources for the annotations of semantic and discourse information concerning aspects of Chinese language characteristics. For example, topic chain and animacy information annotations have never been annotated in previous Chinese cor-

[1]The corpus is available at: https://github.com/fivehills/CCTRS-corpus-/tree/main.

ID	EDU	grammatical structure	tree structure	discourse relation labels	relation distance	verb valency	topic distance	animacy
95-1	飞机飞过	SV	1*2	cu	1	1	NA	NA
95-2	飞机拖白带	SV	2*3	pr	1	2	2	NA
95-3	不敢	VP	1-3*4-7	ce	3	0	NA	NA
95-4	天被隔开	SV	4*5	cj	1	1	3	NA
95-5	天裂	SV	4-5*6	ce	2	1	NA	NA
95-6	漏水	VP	6*7	le	1	1	NA	ina
95-7	鱼不见了	NA	NA	NA	1	0	NA	nonhum

Table 2: Sample annotation sheet.

indicators	segments	RST spans	semantic relations	animacy	action
agreement	.92	.93	.92	.99	.98
Kappa	.82	.83	.81	.87	.86

Table 3: Consistency of annotation by the three annotators.

Corpus	Style	Genre	Multilayer annotations
CDTB (Zhou and Xue, 2015)	PDTB	newpaper	NO
CUHK (Zhou et al., 2014)	PDTB	newpaper	NO
TED-CDB (Long et al., 2020)	PDTB	TED talks	NO
MCDTB (Jiang et al., 2018)	RST	newspaper	NO
CCTRS (ours)	RST& PDTB	fiction	YES

Table 4: comparison of the CCTRS to related Chinese discourse datasets.

pora but they are closely related with discourse relations. Topic chain and animacy are closely related zero amphora (co-reference) (Kong et al., 2019). Moreover, although Jiang et al. (2018) annotated the macro discourse information in Chinese discourse using RST tree structure they adopted different tags from both RST tags and PDTB tags. In this way, it could be a little difficult to compare their tags with other similar discourse corpora (including discourse corpora in other languages). Our CCTRS used PDTB tags so that we can easily compare with either RST-style or PDTB-style corpora.

6 Benchmark for Discourse Parsers

We further apply the CCTRS as a benchmark for comparing and evaluating discourse parsers. For the 500 texts in the CCTRS, 71 are used for development set and 73 for test set, and the remaining 356 texts for training. We implement two parsing sub-tasks. RST parsing usually includes the following sub-tasks: span prediction, nuclearity indication, and relation classification. As mentioned above, there are two different kinds of nodes in the RST tree: nucleus and satellite. The nuclearity indication task aims to predict the nucleus or satellite given two EDUs or spans. However, our corpus does not contain the information on nuclearity. This way, the subtask of nuclearity indication was not be included in RST tree building. Additionally, we used PDTB tags to annoate discours relations. As a result, we can divide RST building into two tasks: span prediction and PDTB relation recognition.

6.1 Methods

We applied the standard micro averaged F1 scores on *Span (Sp.)* between EDUs, and *discourse relation (Rel.)*. The micro-averaged F-1 scores over labelled attachment decisions is applied to make valid comparison (Morey et al., 2017). Span describes the accuracy of RST tree structure construction, while discourse relation assesses the ability to categorize the discourse relations. A typical RST parser also needs to distinguish nuclearity and satellite. However, we did not require this task. Following previous PDTB relation recognition studies, we adopted the 13 relations defined in Table 2. We modified three typical RST parsers recently developed using three different methods: bottom-up (Feng and Hirst, 2014), top-down (Zhang et al., 2020), and LSTM (Koto et al., 2021). The three methods were employed to test the feasibility of our corpus.

6.2 Results

After using three RST parsers in the training data (356 texts), we required the three parsers to recognize span and discourse relations. Table 5 shows the average performance of the three RST parsers on development/test data. The human agreement from the annotators is presented for comparison. It seems that Koto et al. (2021) gets better performance than the other two systems. However, the three parsers perform quite similarly in span and relation recognition. According to the performance

Method	Sp	Rel	F-1
Feng and Hirst (2014)	63.2	43.6	42.9
Zhang et al. (2020)	62.8	44.2	43.1
Koto et al. (2021)	64.3	45.1	43.6
human	83	81	71.3

Table 5: Results over the test set calculated using micro-averaged F-1.

data, we can judge that the performance in the three parsers is very stable in analying the CCTRS. We can also see that human performance is still much higher than the three parsers, meaning there is large space for improvement in future work. Overall, due to the performance, we can conclude that the CCTRS is highly capable of using as a benchmark of discourse parsing.

7 Conclusion

"Run-on" sentences are both typical of and prevalent in Chinese discourse. This study collected 500 "run-on" sentences that were annotated at different levels. The main idea is to integrate the strengths of RST and PDTB with the Chinese discourse characteristics. This paper presented the annotation framework, construction workflow and statistics. Further, this multilayer discourse corpus can serve as an evaluating benchmark of realistic Chinese discourse parsing. Moreover, the CCTRS can provide us with valuable language sources to explore in computational linguistics and linguistics, such as co-referential zero anaphora resolution, and predicting semantic leaps in sentence.

Acknowledgments

This study was supported by *China Postdoctoral Science Foundation* (Outstanding Fund No. `2018T110581`) We appreciate helps provided by Prof. Haitao Liu at Zhejiang University, China. Many thanks go to Ms. Yiyun Zhou, and Ms. Mengjie Xu, and Dr. Yi Shan for their helps in text processing and annotations.

References

Nicholas Asher, Nicholas Michael Asher, and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Emmon Bach. 1986. The algebra of events. *Linguistics and Philosophy*, pages 5–16.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer.

Yuen Chao. 1968. *A grammar of spoken Chinese*. Univ of California Press.

Chauncey Cheng-Hsi Chu. 1998. *A discourse grammar of Mandarin Chinese*. Peter Lang Pub Incorporated.

Bernard Comrie. 1989. *Language Universals and Typology (the 2nd edition)*. The University of Chicago Press.

Vera Demberg, Merel CJ Scholman, and Fatemeh Torabi Asr. 2019. How compatible are our discourse annotation frameworks? insights from mapping rst-dt and pdtb annotations. *Dialogue & Discourse*, 10(1):87–135.

Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165:113679.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521.

Song Gao, Hongxin Zhang, and Haitao Liu. 2014. Synergetic properties of chinese verb valency. *Journal of Quantitative Linguistics*, 21(1):1–21.

Barbara J Grosz, Scott Weinstein, and Aravind K Joshi. 1995. Centering: a framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2014. Using discourse structure improves machine translation evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 687–698.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, et al. 2017. Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv preprint arXiv:1711.05073*.

Jack Hobbs. 1979. Coherence and coreference. *Cognitive Science*, 3:67–90.

Ming Hu and Jingsong. 1989. liú shuǐ jù chū tàn [a study of "run-on" sentence]. *Language Teaching and Research*, (4):42–54.

Bo Huang and Xu Xiao. 2016. *Xiandai Hanyu [Modern Chinese] (4th Edition)*. Higher Education Press.

Feng Jiang, Sheng Xu, Xiaomin Chu, Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2018. Mcdtb: a macro-level chinese discourse treebank. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3493–3504.

Christopher T Kello, Gordon DA Brown, Ramon Ferrer-i Cancho, John G Holden, Klaus Linkenkaer-Hansen, Theo Rhodes, and Guy C Van Orden. 2010. Scaling laws in cognitive sciences. *Trends in cognitive sciences*, 14(5):223–232.

Fang Kong, Min Zhang, and Guodong Zhou. 2019. Chinese zero pronoun resolution: A chain-to-chain approach. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 19(1):1–21.

Fajri Koto, Jey Han Lau, and Timothy Baldwin. 2021. Top-down discourse parsing via sequence labelling. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 715–726.

Klaus Krippendorff. 1980. Validity in content analysis.

Wendan Li. 2005. *Topic chains in Chinese: A discourse analysis and applications in language teaching*, volume 57. Lincom Europa.

Wanqiu Long, Bonnie Webber, and Deyi Xiong. 2020. Ted-cdb: A large-scale chinese discourse relation dataset on ted talks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2793–2803.

Shu Lu and De Zhu. 1979. *Yufa xiuci jianghua [Lectures on grammar and rhetoric]*. Commercial Press.

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

Carey Mcintosh. 1977. A matter of style: stative and dynamic predicates. *PMLA*, 92(1):110–121.

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2017. How much progress have we made on rst discourse parsing? a replication study of recent results on the rst-dt. In *Conference on Empirical Methods on Natural Language Processing (EMNLP 2017)*, pages pp–1330.

Jia Shen. 2012. "lingju" he "liushuiju"—wei chao yuanren xiansheng dancheng 120zhounian erzuo ["minor clauses" and "flowing sentence"—in memorial of 120 birthday of mr. chao yuanren]. *Studies of the Chinese Language*, (5):403–415.

Cai Sheng. 2016. *xiàn dài hàn yǔ liú shuǐ jù yán jiū [A Study of Modern Chinese "Run-on" Sentences]*. Ph.D. thesis, Jinlin University.

Sidney Siegel and N John Castellan. 1981. *JR.(1988): Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill.

Manfred Stede and Arne Neumann. 2014. Potsdam commentary corpus 2.0: Annotation for discourse research. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 925–929.

Kun Sun. 2019. The integration functions of topic chains in chinese discourse. *Acta Linguistica Asiatica*, 9(1):29–57.

Kun Sun and Xiaofei Lu. 2022. Predicting native chinese readers' perception of sentence boundaries in written chinese texts. *Reading and Writing*, pages 1–22.

Kun Sun and Wenxin Xiong. 2019. A computational model for measuring discourse complexity. *Discourse Studies*, 21(6):690–712.

Kun Sun and Lili Zhang. 2018. Quantitative aspects of pdtb-style discourse relations across languages. *Journal of Quantitative Linguistics*, 25(4):342–371.

Feng-Fu Tsao. 1979. *A functional study of topic in Chinese: the first step towards discourse analysis*. Student Book.

Wen Wang and Yong Zhao. 2017. lùn hàn yǔ liú shuǐ jù de jù lèi shǔ xìng[on the sentence class properties of chinese "run-on" sentences. *Chinese Teaching in the World*, 31(2):171–180.

Bonnie Webber. 2004. D-ltag: extending lexicalized tag to discourse. *Cognitive Science*, 28(5):751–779.

Bonnie Webber, Rashmi Prasad, Alan Lee, and Aravind Joshi. 2019. The penn discourse treebank 3.0 annotation manual. *Philadelphia, University of Pennsylvania*.

Nianwen Xue and Yaqin Yang. 2011. Chinese sentence segmentation as comma classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 631–635.

Amir Zeldes. 2017. The gum corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Longyin Zhang, Yuqing Xing, Fang Kong, Peifeng Li, and Guodong Zhou. 2020. A top-down neural architecture towards text-level parsing of discourse rhetorical structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6386–6395.

Lanjun Zhou, Binyang Li, Zhongyu Wei, and Kam-Fai Wong. 2014. The cuhk discourse treebank for chinese: Annotating explicit discourse connectives for the chinese treebank. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 942–949.

Yuping Zhou and Nianwen Xue. 2015. The chinese discourse treebank: A chinese corpus annotated with discourse relations. *Language Resources and Evaluation*, 49(2):397–431.

Appendix

A. Examples

[Example 1]
(1) 夏天义是进了夏天智家的院子,
Xia Tianyi enter PFV Xia Tianzhi's yard,
(2) 我没有进去,
I Not enter,
(3) 只听见白雪的孩子一声比一声尖着哭,
only hear Baixue's kids one M louder than cry
(4) 原本天上还是铁锈红的云,
previously sky up already rust-red clouds,
(5) 一时间黑气就全罩了。
one time black air all over the cover PFV
[Free translation:] Xia Tianyi walked into Xia Tianzhi's courtyard, but I did not go in, and then I only heard the sound of the kid crying, and crying louder and louder. Strangely enough, when raising my head, I found that the original sky is still rust-red clouds, a time when the black gas is all over.

[Example 2]
(1) 河边站满了人,
river stand fully people
(2) 四只朱色长船在潭中滑着,
four M red long boats in river floating
(3) 龙船水刚刚涨过,
dragon boats water just rise
(4) 河中水皆豆绿,
river middle water all green
(5) 天气又那么明朗,
weather so sunny
(6) 鼓声蓬蓬响着,
drums sounds PFV
(7) 翠翠抿着嘴一句话不说,
Cuicui purse PFV one word Not say
(8) 心中充满了不可言说的快乐。
in heart filled with indescribable happiness
[Free translation] The river was full of people (1), four long vermilion boats were sliding in the pool(2), the water of the dragon boat had just risen(3), the water in the river was all bean green(4), the weather was so bright(5), the drums were sounding(6), [`semantic leap`] Cuicui pursed her lips without saying a word(7), her heart was full of unspeakable joy(8).

[Example 3]
Jacobs Engineering Group Inc.'s Jacobs International unit was selected to design and build a microcomputer-systems manufacturing plant in County Kildare, Ireland, for Intel Corp. Jacobs is an international engineering and construction concern. [`NoRel`] Total capital investment at the site could be as much as $400 million, according to Intel. (WSJ_1081)

[Example 4]
(1) 这时候一架飞机飞过,
this time a M airplane flew over
(2) 飞机后拖了条白带,
airplane behind trailed PVF M white band
(3) 经久不散,

long-standing,
(4) 天就被割开了,
sky Bei cut open PFV
(5) 或者是天裂了,
or sky cracked PFV
(6) 漏了水,
leaked PFV water
(7) 鱼也不见了。
fish also disappeared.
[Free translation] At this time an airplane flew over, and a white band trailed behind the airplane for a long time, and the sky was cut open, or the sky cracked and leaked, [`semantic leap`] and the fish disappeared.

[Abbreviations of glosses]
`PFV` – perfective aspect
`M` – measure word

B. Figures

In this section there are three figures.

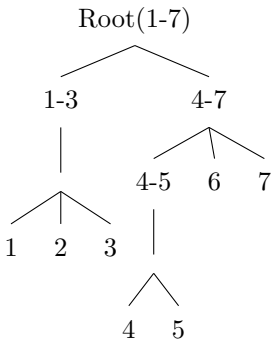


Figure 1: An RST constituent tree for Example 4 and Table 2 in the CCTRS

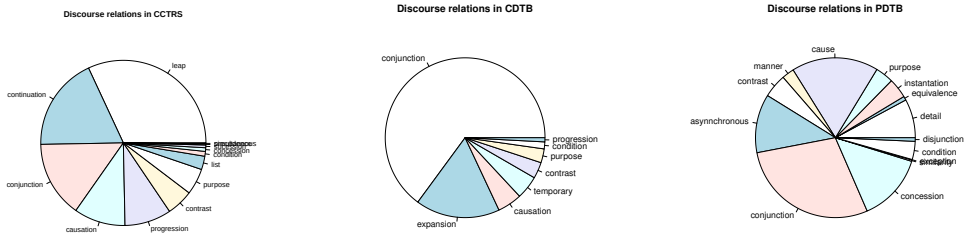


Figure 2: Proportions of discourse relations in the three discourse corpora

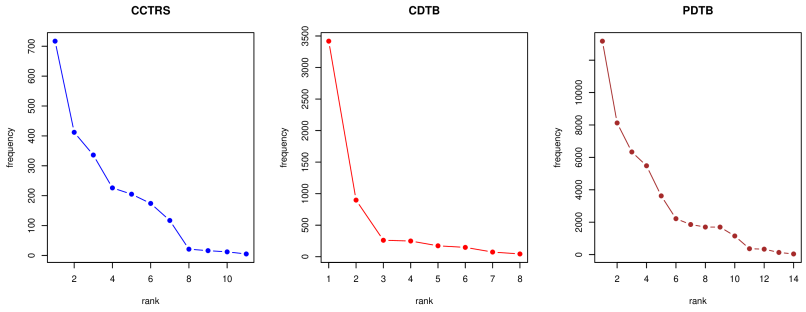


Figure 3: Frequency distribution of semantic relations in the three discourse corpora

Utilizing BERT Intermediate Layers for Unsupervised Keyphrase Extraction

Mingyang Song, Yi Feng and Liping Jing*
Beijing Key Lab of Traffic Data Analysis and Mining
Beijing Jiaotong University, China
mingyang.song@bjtu.edu.cn

Abstract

Extensive pre-trained language models such as the transformer-based BERT have been compelling at language modeling, achieving impressive results on numerous natural language downstream tasks. It has also been demonstrated that they implicitly retain factual knowledge in their parameters after pre-training. Understanding what the pre-training procedure of language models learns is critical to utilizing and enhancing them for Unsupervised Keyphrase Extraction (UKE). However, most existing BERT-based studies about UKE only use the single intermediate layer of BERT (e.g., the last layer) and ignore the latent knowledge in the intermediate layers. Therefore, in this paper, we analyze and explore the potential of utilizing BERT intermediate layers to enhance text representations and improve the performance of the state-of-the-art BERT-based unsupervised keyphrase extraction model. Specifically, we first verify and analyze the effect of adopting different BERT intermediate layers on the recent state-of-the-art unsupervised keyphrase extraction model. Then, based on the analysis, we propose a simple and effective feature aggregation strategy. Experimental results on several benchmark datasets demonstrate the effectiveness of aggregating intermediate layers of BERT to enhance text representations on the unsupervised keyphrase extraction task.

1 Introduction

Keyphrase extraction (KE) aims to extract a set of words or phrases from a given document that represents the salient information and main topics of the document (Hasan and Ng, 2014). KE models typically can be divided into supervised and unsupervised. Supervised approaches (Song et al., 2021) need large-scale annotated training data and perform poorly when transferred to different domain or type datasets. Unsupervised keyphrase extraction (UKE) approaches (Mihalcea and Tarau, 2004; Liang et al., 2021) are more universal and adaptive by extracting phrases based on information from the source document itself than the supervised method. This paper focuses on the unsupervised keyphrase extraction model.

A critical breakthrough in natural language processing is the use of heavily pre-trained transformers for natural language modeling, such as BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019). These Pre-trained Language Models (PLMs) are powerful for many downstream tasks in natural language processing and information retrieval, which have thus become an essential part in most cases. Therefore, research has also been done on BERT, especially to reveal what linguistic information is available in different parts of the model (Jawahar et al., 2019; de Vries et al., 2020). It has been noted that BERT progressively learns linguistic information roughly in the same order as the classic language processing pipeline: surface features are expressed in lower intermediate layers, syntactic features more in middle intermediate layers, and semantic ones in higher intermediate layers. Based on the above phenomenon, much recent work (Song et al., 2020, 2022) focuses on exploring the potential of utilizing BERT intermediate layers to enhance the fine-tuning performance of BERT. They demonstrated that each layer has different specializations, so combining information from different layers may be more beneficial instead of selecting a single one based on the best overall performance.

With the development of the pre-trained language models, recent unsupervised keyphrase extraction approaches (Sun et al., 2020; Liang et al., 2021; Ding and Luo, 2021) adopt the last layer of BERT as the embedding layer to obtain phrase and document representations instead of using the traditional pre-trained word vector, which significantly improves the performance of unsupervised

*Corresponding author.

keyphrase extraction. However, as mentioned earlier, the pre-trained language models (e.g., BERT (Devlin et al., 2019)) store rich language knowledge in the intermediate layers. Therefore, only using the single layer of BERT wastes the latent knowledge hidden in BERT. Meanwhile, judging a candidate phrase as a keyphrase also requires considering various features of natural languages (Hasan and Ng, 2014; Song et al., 2021), such as syntax, semantics, etc.

Motivated by the above phenomenon, we first probe the effectiveness of the intermediate layers of the pre-trained language model BERT on unsupervised keyphrase extraction. Then, we investigate several feature integration strategies for aggregating the middle layers of BERT to improve the performance of the state-of-the-art baseline (JoingGL (Liang et al., 2021)). Experimental results on DUC2001, Inspec, and SemEval2010 datasets show that combining intermediate layers of BERT as the embedding layer obtains better performance than using the single one.

2 Methodology

Given the sequence $\mathbf{x} = \{x_1, x_2, ..., x_m, ..., x_M\}$ with M tokens, we adopt BERT to encode it and obtain the hidden states for the i-th token from all L intermediate layers, $\mathbf{h}_i^L = \{h_i^1, h_i^2, ..., h_i^L\}$. Typically, L is set to 12. In this paper, based on the recent state-of-the-art model (JointGL (Liang et al., 2021)), we first probe the performance of different intermediate layers of BERT as the embedding layer on the current state-of-the-art model JointGL. Then, we test several feature integration strategies for combining the middle layers of BERT as the embedding layer on the baseline JointGL.

2.1 A Model-dependent Analysis

Existing embedding-based unsupervised keyphrase extraction models rely on the pre-trained language models (e.g., BERT) to achieve significant progress. Still, there is no work to probe in detail the impact of the features of different BERT intermediate layers on the unsupervised keyphrase extraction task. Therefore, to better utilize the pre-trained language models, we give a model-dependent analysis. Figure 1 shows the effectiveness of the baseline model JointGL by adopting different intermediate layers as the embedding layer on DUC2001, Inspec, and SemEval2010 datasets.

First of all, as can be seen from the results, the last layer of BERT is not always the one that gives the best performance of keyphrase extraction. Second, we found that the dependence on natural language features is different for different keyphrase extraction datasets when obtaining text representation. For DUC2001 and SemEval2001, the syntactic information in the middle intermediate layer is more critical. For Inspec, surface information is more critical. The above phenomenon is that the choice of features at different intermediate layers of BERT determines the effectiveness of phrases and documents representation. Overall, keyphrases in various datasets may rely on different linguistic features. Finally, an interesting phenomenon is that the 11-th intermediate layer achieves the worst F1, precision, and recall results. It is interesting to investigate why the 11-th layer obtains such results in future work.

2.2 Selective Feature Aggregation

We introduce a simple selective feature aggregation strategy to comprehensively use the linguistic knowledge stored in the pre-trained language models. Based on our model-dependent analysis, the intermediate layers with the higher evaluation metrics are selected as candidate layers in the first step. We generally choose the top K scores with their corresponding layers. For the second step, we use a weighted pooling operation to integrate the selected K layers as follows:

$$h_i^{'} = \gamma h_i^{k=1} + (1-\gamma) \sum_{k=2}^{K} h_i^k \qquad (1)$$

where γ is the balance factors and $h_i^{'}$ indicates the final representation of i-th input token. Through the above strategy, different feature information contained in different layers in BERT is fused to assist unsupervised keyphrase extraction.

3 Experiments

3.1 Datasets

To better verify the effectiveness of the proposed strategy, we evaluate our model on three benchmark keyphrase extraction datasets: DUC2001 (Wan and Xiao, 2008), Inspec (Hulth, 2003), and SemEval2010 (Kim et al., 2010).

3.2 Evaluation

We follow the common practice and evaluate the performance of our models in terms of f-measure

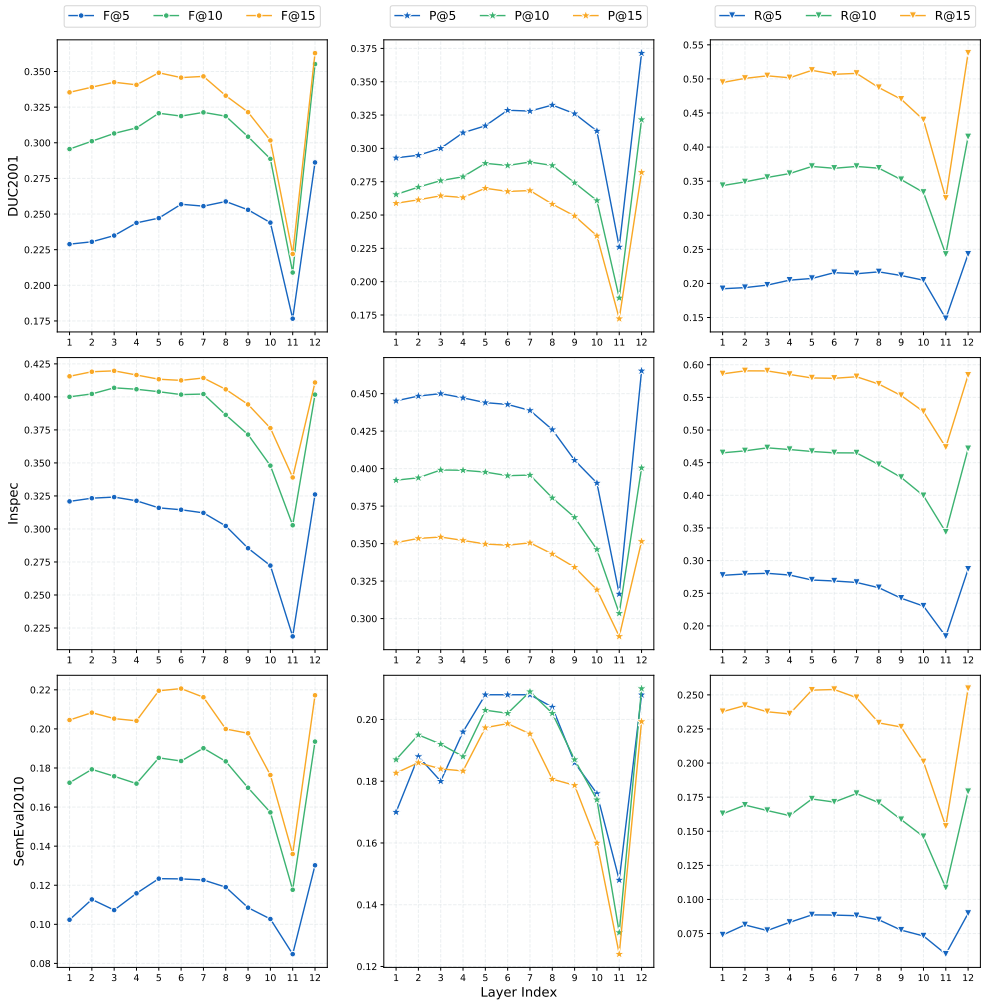


Figure 1: Results of the different intermediate layers of BERT for the baseline model JointGL on DUC2001, Inspec, and SemEval2010 test sets. The x-axis represents the index of the intermediate layers of BERT.

at the top N keyphrases (F1@N), and apply stemming to both extracted keyphrases and gold truth. Specifically, we report F1@5, F1@10 and F1@15 of each model on three benchmark keyphrase extraction datasets.

3.3 Implementation Details

We follow the previous baseline work (Liang et al., 2021) and adopt the same settings on the DUC2001, Inspec, and SemEval2010 datasets. For the selective layer aggregation strategy, γ is set to 0.95, 0.9, and 0.5 for DUC2001, Inspec, and SemEval2010. Specifically, K is set to 4, which means four intermediate layers with higher evaluation scores (F@5) are used in our strategy. Therefore, the $12, 8, 6, 7$-th intermediate layers are selected for DUC2001; the $12, 3, 2, 1$-th intermediate layers are selected for Inspec; the $12, 5, 6, 7$-th intermediate layers are selected for SemEval2010 (as shown in Table 1).

3.4 Results

Table 1 shows the results of F1, precision, recall, and @5, 10, and 15 using the embedding-based baselines and JointGL with different layer aggregation strategies on all datasets.

From the results in Table 1, we can see that the proposed feature aggregation strategy outperforms existing embedding-based unsupervised keyphrase extraction methods in most cases. The main reason is that the two most essential procedures of unsupervised keyphrase extraction methods are text representation and semantic similarity calculation. Our strategy obtains more comprehensive phrase and document representations by considering different linguistic knowledge stored in the pre-trained language models, which naturally improves the performance of the keyphrase extraction model.

Compared with different layer aggregation methods, our method has achieved significant improve-

Model	DUC2001			Inspec			SemEval2010		
	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15
Statistical Models									
TF-IDF (Jones, 2004)	9.21	10.63	11.06	11.28	13.88	13.83	2.81	3.48	3.91
YAKE (Campos et al., 2018)	12.27	14.37	14.76	18.08	19.62	20.11	11.76	14.4	15.19
Graph-based Models									
TextRank (Mihalcea and Tarau, 2004)	11.80	18.28	20.22	27.04	25.08	36.65	3.80	5.38	7.65
SingleRank (Wan and Xiao, 2008)	20.43	25.59	25.70	27.79	34.46	36.05	5.90	9.02	10.58
TopicRank (Bougouin et al., 2013)	21.56	23.12	20.87	25.38	28.46	29.49	12.12	12.90	13.54
PositionRank (Florescu and Caragea, 2017)	23.35	28.57	28.60	28.12	32.87	33.32	9.84	13.34	14.33
MultipartiteRank (Boudin, 2018)	23.20	25.00	25.24	25.96	29.57	30.85	12.13	13.79	14.92
Embedding-based Models									
EmbedRankd2v (Bennani-Smires et al., 2018)	24.02	28.12	28.82	31.51	37.94	37.96	3.02	5.08	7.23
EmbedRanks2v (Bennani-Smires et al., 2018)	27.16	31.85	31.52	29.88	37.09	38.40	5.40	8.91	10.06
SIFRank (Sun et al., 2020)	24.27	27.43	27.86	29.11	38.80	39.59	-	-	-
SIFRank+ (Sun et al., 2020)	30.88	33.37	32.24	28.49	36.77	38.82	-	-	-
KeyGames (Saxena et al., 2020)	24.42	28.28	29.77	32.12	40.48	40.94	11.93	14.35	14.62
AttentionRank (Ding and Luo, 2021)	-	-	-	24.45	32.15	34.49	11.39	15.12	16.66
MDERank (Zhang et al., 2021)	23.31	26.65	26.42	27.85	34.36	36.40	13.05	18.27	20.35
JointGL (Liang et al., 2021)	28.62	35.52	36.29	32.61	40.17	41.09	13.02	19.35	21.72
JointGL (Using the First Layers)	22.88	29.51	33.34	31.91	39.81	41.57	10.76	17.46	20.38
JointGL (Sum the 1-4 Layers)	23.60	30.44	33.78	32.06	40.33	41.64	11.63	17.73	20.28
JointGL (Sum the 5-8 Layers)	25.96	32.53	34.94	30.60	39.81	40.92	13.00	19.31	21.36
JointGL (Sum the 9-12 Layers)	24.31	28.26	29.94	27.25	34.71	37.56	10.66	15.64	17.69
JointGL (Sum the 1-12 Layers)	25.57	32.03	34.09	30.22	39.25	40.43	11.98	17.67	20.57
JointGL (Selective Feature Aggregation)	**28.92**	**35.71**	**36.54**	**32.40**	**40.71**	**41.92**	**13.25**	**19.93**	**22.23**

Table 1: Performance on DUC2001, Inspec and SemEval2010 test sets. F1 scores on the top 5, 10, and 15 keyphrases are reported. The best results of our model are bolded in the table.

ments, demonstrating the effectiveness of the proposed layer aggregation strategy.

Furthermore, it can be seen that different layer integration strategies have different effects on different datasets, which also shows the importance of potential knowledge mining in the pre-trained language model BERT.

4 Related Work

Unsupervised keyphrase extraction models mainly can be grouped into the traditional models (Jones, 2004; Mihalcea and Tarau, 2004; Bougouin et al., 2013) and embedding-based models (Sun et al., 2020; Bennani-Smires et al., 2018; Saxena et al., 2020; Liang et al., 2021). With the proposal and vigorous promotion of pre-trained language models, language models have become the backbone of most downstream natural language processing tasks, and significant progress has been made. Recent unsupervised keyphrase extraction models (Sun et al., 2020; Ding and Luo, 2021; Liang et al., 2021) adopt the last intermediate layer of the pre-trained language models as the embedding layer.

Different from the previous studies, this paper focuses on probing and aggregating the intermediate layers of the pre-trained language models for improving the performance of UKE.

5 Conclusions and Future Work

In this paper, we probe and analyze the effectiveness of aggregating the intermediate layers of BERT for unsupervised keyphrase extraction. To the best of our knowledge, we are the first work to probe BERT for unsupervised keyphrase extraction. Based on the findings of our non-parametric probing task, we propose a simple and effective feature integration strategy, which combines the intermediate layers of the pre-trained language models to improve the performance of UKE.

The main goal of this paper is to provide an empirical study of the existent models. Since we do not propose new models, there are no potential social risks to the best of our knowledge. Our work may benefit the research community by providing more introspection to the current state-of-the-art keyphrase extraction models. In future work, extending our work for self-supervised keyphrase extraction can also provide more insights into the utility of BERT for keyphrase extraction. Furthermore, it will be interesting to investigate why using the 11-th intermediate layer as the embedding layer leads the performance collapse.

References

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In *CoNLL*, pages 221–229. Association for Computational Linguistics.

Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. In *NAACL-HLT (2)*, pages 667–672. Association for Computational Linguistics.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *IJCNLP*, pages 543–551. Asian Federation of Natural Language Processing / ACL.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *ECIR*, volume 10772 of *Lecture Notes in Computer Science*, pages 806–810. Springer.

Wietse de Vries, Andreas van Cranenburgh, and Malvina Nissim. 2020. What's so special about bert's layers? a closer look at the nlp pipeline in monolingual and multilingual models. In *EMNLP (Findings)*, pages 4339–4350. 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 *NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics.

Haoran Ding and Xiao Luo. 2021. Attentionrank: Unsupervised keyphrase extraction using self and cross attentions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1928.

Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *ACL (1)*, pages 1105–1115. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *ACL (1)*, pages 1262–1273. The Association for Computer Linguistics.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *EMNLP*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL (1)*, pages 3651–3657. Association for Computational Linguistics.

Karen Spärck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60(5):493–502.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *SemEval@ACL*, pages 21–26. The Association for Computer Linguistics.

Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. 2021. Unsupervised keyphrase extraction by jointly modeling local and global context. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 155–164, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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. *CoRR*, abs/1907.11692.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *EMNLP*, pages 404–411. ACL.

Arnav Saxena, Mudit Mangal, and Goonjan Jain. 2020. Keygames: A game theoretic approach to automatic keyphrase extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2037–2048.

Mingyang Song, Yi Feng, and Liping Jing. 2022. Hyperbolic relevance matching for neural keyphrase extraction.

Mingyang Song, Liping Jing, and Lin Xiao. 2021. Importance Estimation from Multiple Perspectives for Keyphrase Extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Youwei Song, Jiahai Wang, Zhiwei Liang, Zhiyue Liu, and Tao Jiang. 2020. Utilizing bert intermediate layers for aspect based sentiment analysis and natural language inference. *CoRR*, abs/2002.04815.

Yi Sun, Hangping Qiu, Yu Zheng, Zhongwei Wang, and Chaoran Zhang. 2020. Sifrank: A new baseline for unsupervised keyphrase extraction based on pre-trained language model. *IEEE Access*, 8:10896–10906.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, pages 855–860. AAAI Press.

Linhan Zhang, Qian Chen, Wen Wang, Chong Deng, Shiliang Zhang, Bing Li, Wei Wang, and Xin Cao. 2021. Mderank: A masked document embedding rank approach for unsupervised keyphrase extraction. *CoRR*, abs/2110.06651.

"Der Frank Sinatra der Wettervorhersage": Cross-Lingual Vossian Antonomasia Extraction

Michel Schwab[1], **Robert Jäschke**[1,2] **and Frank Fischer**[3]
[1]Humboldt-Universität zu Berlin, Germany
[2]L3S Research Center, Hannover, Germany
[3]Freie Universität Berlin, Germany
{michel.schwab,robert.jaeschke}@hu-berlin.de
fr.fischer@fu-berlin.de

Abstract

We present a cross-lingual approach for the extraction of Vossian Antonomasia, a stylistic device especially popular in newspaper articles. We evaluate a zero-shot transfer learning approach and two approaches that use machine-translated training and test data. We show that our proposed models achieve strong results on all test datasets in the target language. As annotated data is sparse, especially in the target language, we generate additional test data to evaluate our models and conclude with a robustness study on real-world data.

1 Introduction

Automatic detection and extraction of stylistic devices is an important task for understanding natural language, especially for understanding figurative language. However, most research focuses on English corpora, since labeled data is, to our knowledge, only available in English. Often those approaches cannot be applied to other languages for various reasons, for example, the lack of labeled data, syntactic variations, or semantic differences. In this paper, we study the cross-lingual extraction of the stylistic device Vossian Antonomasia (VA).

VA is a specific kind of antonomasia closely related to metonymies and metaphors. In short, it is used to describe an entity (target) by mentioning another entity (source) and a context (modifier) that refers to the target. Usually, the source entity is famous and well-known to the reader in order to understand the author's intentions. Particularly, a set of characteristics of the source is used implicitly to describe the target. The modifier shifts the source's characteristics to the target's environment. The title of this paper may serve as an example. In a German newspaper article (elm, 2013) Elmar Gunsch (nicknamed "Die Stimme"[1]) is called the "Frank Sinatra of weather forecasting". It is also explained why: "Elmar Gunsch's sonorous bass brought glamor to ill-humored German television in the seventies." The popular American singer and actor Frank Sinatra serves as the source of this VA. Through the formulation, some of Sinatra's characteristics (voice, entertainment qualities, and popularity) are transferred to the target, Elmar Gunsch. The dependent genitive construction ("of weather forecasting") represents the modifier.

The automated detection and extraction of VA is a non-trivial task as the syntax can be ambiguous. Take, for instance, "the Galileo of welfare reform" (Roberts, 1992) vs. "the Galileo of the Fantasy line" (gal, 1990). While the first example is a VA expression referring to a senator, the latter is the name of a cruise from a cruise line company. This is also one of the reasons why rule-based approaches fail, as seen in Schwab et al. (2019) where a trained neural network outperforms the rule-based approaches. While the extraction of VA from English corpora has already been studied Schwab et al. (2019, 2022), there are no studies on automated approaches to find VA in other languages, yet. This is the starting point for this paper: we study the automated extraction of VA in another language – German.

As the lack of annotated data is one of the main problems for the study in different languages, we will use three different sequence tagging approaches: The first is based on zero-shot transfer learning, the second and third on machine translation and word alignments of the annotated data. All models employ pre-trained language models because they achieved the best results on English VA extraction, see Schwab et al. (2022).

Despite the sparse occurrence of VA in text corpora, it can assist several NLP applications. Co-reference resolution can be supported as the source entity should not be a single co-reference chain but together with the modifier part of the target chain. The generation of fruitful and interesting

[1]"The Voice", our translation

Lanz, der OBAMA des *deutschen Fernsehens* **Lanz**, the OBAMA of *German television*
Ein *spanischer* LIONEL JOSPIN müsse her. A *Spanish* LIONEL JOSPIN was needed.
Statine, der BENTLEY unter *den Kardioprotektiva* **Statins**, the BENTLEY of *cardioprotectants*.

Table 1: Three examples of German VA expressions together with their translation and alignment.

text is another reason, especially the generation of headlines. It also assigns attributes to entities and connects entities together which can lead to interesting question-answering tasks. In general, it is a use case for similar cases where there is a lack of large annotated datasets and can therefore show ways to tackle similar downstream tasks that are not as rich as common NLP tasks as named entity recognition.

In the following, we want to answer whether 1) our models can compete with mono-lingual approaches, 2) machine translation based models can compete against new zero-shot models, and 3) our models are robust against real-world data. Our code and data are freely available.[2]

2 Related Work

The automated extraction and detection of VA has not been studied deeply. There exist rule-based approaches (Fischer and Jäschke, 2019; Schwab et al., 2019), the latter also trained a neural network based on non-contextualized word-embedding and bi-directional LSTM layers to classify sentences into whether they include VA expressions or not. Schwab et al. (2022) constructed neural network models from scratch but their best models are based on pre-trained language models, BERT (Devlin et al., 2018). In addition to a binary sentence classifier, the authors tackled the problem of detecting all chunks of a VA expression inside a sentence. They created an annotated dataset and showed that their models are robust on real-world data. They also showed that adding unlabeled random sentences to their training data as negative instances improved the model. Adding such sentences helped to diversify the syntactic variations of negative instances and also generated a class imbalance that is closer to real-world data without the cost of annotation.

Cross-lingual approaches for detecting similar stylistic devices, for instance, cross-lingual metaphor detection, have also not been studied deeply: Tsvetkov et al. (2014) used lexical semantic features and word embeddings (ENG-{ESP, FAS, RUS}), whereas Víta (2020) studied the problem of cross-lingual metaphor paraphrase detection (ENG-CZE). They used machine translation, but also multilingual word embeddings, MUSE (Lample et al., 2018). Aghazadeh et al. (2022) presented models that employ the layers of pre-trained language models in a zero-shot probing scenario.

[2] https://vossanto.weltliteratur.net/

training data		test data	
before	after	before	after
the	of	die/der/das	der/des
a/an	for	ein/e	von/vom
	among	diese/r/m	unter
		Art	aus
		als	für
		den/dem	-

Table 2: Most common syntactic variations around the source in the training and test data. The column 'before' shows the words appearing before, 'after' the words following the source in the sentence. '-' signals that there is no boundary word after the source but either the end of the sentence or the modifier.

3 Datasets

In this section, we describe the datasets and the annotation process. The training data is in English whereas the test data consist of German texts. Comparing the diversity of the datasets, we can clearly see that the test data is richer in terms of the syntactic variations of VA, see Table 2. The table shows the most frequently used boundary words around the source. Whereas the training data has a limited collection of boundary words, the test data shows a great diversity. This is because of the manual collection of one of the test data which did not follow any syntactic restrictions like the training data.

3.1 Training Data

NYT-0: The dataset originally emerged from a semi-automatic rule-based approach from Schwab et al. (2019) on The New York Times Annotated Corpus (Sandhaus, 2008). Schwab et al. (2022) expanded the dataset and tagged all VA chunks inside a sentence on the word level. We updated the target annotations to improve consistency (cf. Section 3.3) and corrected mislabeled annotations. The dataset contains 3,065 sentences with VA expressions (which we call *positive* in the sequel) and 2,930 without (*negative*). All VA expressions in this dataset follow a specific syntax, that is, the/a/an SOURCE of/for/among, see Table 2.

NYT-50: Like Schwab et al. (2022) we add 50,000

random sentences from the New York Times Corpus (Sandhaus, 2008) to the NYT-0 dataset to diversify the negative instances in the dataset and to tackle the biased share of positive instances in the NYT-0 dataset ($\approx 50\%$).

3.2 Test Data

UMBL: This VA collection[3] was manually gathered from German newspaper articles but also in radio, TV, or videos between 2009 and 2014. It only contains sources and modifiers of VA expressions, but not the targets and the original sentences. We tried to collect the sentences from the original articles, but could not use all instances as some of the articles the expression appeared in were behind paywalls or did not exist anymore. Out of 470 positive instances, we could get full information in 362 cases which we annotated as explained in Section 3.3. The collection is not based on any syntactic rules, thus, the VA expressions contain broader syntactic variations compared to the NYT dataset, as can be seen in Table 2. Also, in this dataset the modifier may appear before the source, see Table 1, Ex. 2 which does not appear in the NYT dataset.

ZEIT: A dataset where Jäschke et al. (2017) used a complex rule-based approach to collect VA instances from German weekly "Die Zeit" (covering 1995 to 2011). In total, they found 1,456 candidate sentences of which 224 are positive. The 224 instances together with the sentence they appear in are publicly available and fit as a test dataset. Source and target were already annotated which left us to annotate the modifier.

Generation of negative data: As both datasets only contain positive instances, we generated additional samples that consist of negative instances to evaluate our model accurately. We use the sparseness of the phenomenon to create one random sample and two focused samples that contain instances similar to those in the training data in terms of syntax or the choice of entities to make sure that none of these reasons are biasing our models. Each of those samples includes 3,000 sentences.

NEG1: The dataset contains sentences which include phrases that are syntactically similar to the VA expressions in the NYT-0 dataset regarding source and modifier. That is, the modifier appears behind the source, mostly separated by a delimiter word (e.g., "der/des" - "of"), see Ex. 1, Table 1. So, we extracted all Wikidata entities that have a German label which contains one of the delimiter words between two other words, for example, "Königin von England". We then took a sentence from the corresponding German Wikipedia web page that included the label and use this sentence as an instance in our test dataset.

NEG2: Most of the source entities in all three datasets are humans. Thus, we want to analyze whether the choice of entities in test instances has an impact on the prediction. We use a corpus of a German newspaper, "taz, die tageszeitung" (TAZ) (200, 2005) to create the sample. This corpus contains more than one million German news articles from 1985 to 2005. For each of the 1,691 distinct source entities in the NYT-0, UMBL and ZEIT datasets, we extracted three random sentences in the TAZ corpus that include the entity's name. In total, we could extract 3,940 sentences and again we used a sample of 3,000 sentences. Due to different reasons, for example, different spellings, we could not find sentences for all entities.

NEG3: is generated by a random selection of sentences from the TAZ corpus mentioned above.

3.3 Annotation Process

In the UMBL and ZEIT corpus, we annotated different chunks, whereas in the NYT-0 dataset, we only updated the targets due to consistency, see below. For NEG-1, NEG-2 and NEG-3, all found VA expressions were replaced by negative instances to keep those samples consist without any VA expressions.

We follow the IOB annotation scheme from Schwab et al. (2022) with one exception: For target annotations, we annotate the whole noun phrase instead of the entity only (leaving out relative clauses due to length) as this has not been annotated consistently before. The annotations of NEG1, NEG2, and NEG3 were conducted by one trained annotator. The annotator found 43 VA expressions in NEG-2, 3 in NEG-3 and none in NEG-1. Those were replaced by negative sentences following the generation process for each sample to have consistent negative samples. Additionally, two other trained annotators annotated 100 random instances of each sample for the quality assessment of the annotations which resulted in a full agreement of all instances except one which was discussed and annotated accordingly.

[3] https://umblaetterer.de/datenzentrum/vossianische-antonomasien.html

4 Methods

We model the problem as a sequence tagging task (Schwab et al., 2022): For each word w_i of sentence S predict a tag t_i which indicates whether w_i is part of target, source, or modifier, or not a VA part at all.

We study a zero-shot cross-lingual transfer scenario as well as models that use machine-translated test or training data. The fine-tuning step in all three methods is conducted by adding a linear layer on top of the pre-trained model architecture that outputs a tag for each token of the input.

0shot: Zero-shot approaches on multilingual language models have recently shown great advances. They are often used in cases like ours, that is, there exists no or only few annotated data in the target language. In short, a language model is pre-trained on a multilingual corpus and then fine-tuned only with the annotated data from one language (source language). Without seeing any annotated data from the target language, it has been shown that those fine-tuned models are able to transfer their learning to languages that they have been pre-trained with, see Conneau et al. (2020). We fine-tune the XLM-RoBERTa model (Conneau et al., 2020) with the NYT training data and evaluate it on the test data.

de2en: We translate the German test data to English using machine translation, in particular the FAIRSEQ toolkit (Ott et al., 2019). We align the original translated sentence pairs with a word alignment tool (Jalili Sabet et al., 2020) and project the corresponding tags to the translated sentences.[4] Then, we fine-tune a BERT model (Devlin et al., 2018) with the NYT datasets and evaluate it on the translated and aligned test data.

en2de: We use the architecture as in de2en but the other way round: We translate the training data to German using FAIRSEQ and project the tags with the word aligner.[4] Then, we fine-tune a German pre-trained language model, DBMDZ BERT,[5] with the translated data and evaluate it on the test data.

5 Experiments

5.1 Experimental setup

Hyperparameter optimization is applied on epoch (e), learning rate (lr), and batch size (b) for all models. For the implementation, we use the Hugging Face transformers framework (Wolf et al., 2020).

[4]We post-process the results and automatically correct minor alignment errors, for example, false tag orders.

[5]https://github.com/dbmdz/berts

model	train	test	precision	recall	F_1
0shot	NYT-0	UMBL	0.911	0.675	0.776
		ZEIT	0.926	0.726	0.814
		COMB	0.881	0.695	0.777
	NYT-50	UMBL	0.890	0.306	0.456
		ZEIT	0.867	0.429	0.573
		COMB	0.869	0.354	0.503
de2en	NYT-0	UMBL	0.809	0.599	0.689
		ZEIT	0.822	0.642	0.721
		COMB	0.780	0.616	0.688
	NYT-50	UMBL	0.824	0.550	0.660
		ZEIT	0.836	0.624	0.714
		COMB	0.818	0.579	0.678
en2de	NYT-0	UMBL	0.915	0.835	0.873
		ZEIT	0.931	0.864	0.896
		COMB	0.865	0.846	0.855
		ROB	0.532	1.000	0.695
	NYT-50	UMBL	0.907	0.803	0.852
		ZEIT	0.936	0.843	0.887
		COMB	0.887	0.818	0.851
		ROB	0.574	0.794	0.667

Table 3: Performance on all test datasets using micro-average over all VA tags. "COMB" shows the scores for all test datasets (including NEG1, NEG2 and NEG3) combined. "ROB" shows the scores of the robustness study which is only conducted for the en2de model.

0shot: We fine-tune the XLM-RoBERTa base model that has 12 transformer blocks, 12 attention heads, a hidden size of 768 and ca. 270 million parameters ($e = 4, b = 16, lr = 5 \cdot 10^{-5}$).

de2en: We apply the German-to-English model from Ng et al. (2019) to translate the test datasets using FAIRSEQ. We align the words with SimAlign (Jalili Sabet et al., 2020), a word aligner which uses static and contextualized embeddings. We fine-tune the multilingual BERT base model (Devlin et al., 2018) (mBERT) ($e = 4, b = 64, lr = 3 \cdot 10^{-5}$).

en2de: We use the same tools as for de2en, but apply the English-to-German model from Ng et al. (2019) to translate the training data and apply SimAlign to project annotations. We fine-tune a German model (dbmdz/bert-base-german-cased) ($e = 3, b = 16, lr = 3 \cdot 10^{-5}$).

5.2 Experimental Evaluation

5.2.1 Evaluation on annotated data

Table 3 shows the results. The en2de model outperforms the other two models by large margins on all test datasets. This comes as a surprise, we expected larger errors in the translation and the tag alignments of the training data. The results show only little differences compared with monolingual

approaches from Schwab et al. (2019) and Schwab et al. (2022). This is remarkable as the test datasets are much more diverse in terms of syntax and entity usage. For all models, it holds that they had better results on the ZEIT dataset compared to the UMBL dataset. One reason is the syntactic diversity of VA expressions UMBL contains. Most false negative errors were VA expressions that had different syntactic structure, like Example 2 in Table 1. In the negative samples, en2de predicted most false negative errors, but no more than 61 false tags in all 9,000 instances. Most tags were falsely predicted in sample NEG2. The addition of unlabeled data had a huge impact on the 0shot model where the performance dropped heavily. The other two models showed almost no difference.

5.2.2 Robustness study

We conduct a study of our best performing model, en2de, trained with NYT0 and NYT50, respectively, on a sample of unlabeled real-world data. 1,000,000 random sentences of the TAZ corpus (introduced in Section 3) are tagged by the model. We analyze the predictions following Schwab et al. (2022): For each predicted tag, the tagger returns a prediction score. The tag with the highest score is the prediction. We compute the difference of the highest and second highest score which we interpret as a confidence score for the respective prediction. For all words of a sentence the confidence scores are averaged to represent the confidence of the prediction for a sentence. As in (Schwab et al., 2022), we take the 30 most confident predictions including at least one source and one modifier prediction tag (positive, i.e., a predicted VA expression) and 30 without those predictions (negative), as well as the 30 most unconfident (15 pos, 15 neg) to get the same share in the evaluation which we annotated manually. Table 3 indicates that the model has more problems on tagging real-world texts, it looses around .16 (NYT-0) and .18 (NYT-50) in F_1 compared to the COMB dataset. Still, the results are reasonable referring to the complexity of the task.

5.2.3 Error Analysis

Analyzing the false positive prediction errors of our best model, en2de, it stands out that the model overfits to sentences that show syntactic patterns similar to the syntax of VA expressions in the training data. For example, in the sentence "Mike Stern ist das Raubein unter den Jazzgitaristen."[6] the model falsely tagged 'Mike Stern' as target, 'Raubein' as source and 'Jazzgitaristen' as modifier. A similar example is "AIDS – Super-Gau der Gentechnologie?"[7] where 'Super-Gau' was tagged as source and 'Gentechnologie' as modifier. In both examples, syntax and even semantic dependencies are close to the definition of VA. In particular, a common noun like 'Raubein' (or 'Super-Gau') is used in a specific environment, 'Jazzgitaristen' ('Gentechnologie', respectively). But as those nouns are no named entities and only the literal meaning of the words is used, the phrases cannot be VA expressions.

On the other hand, the false negative errors appeared mainly when the syntax of the VA expressions was new to the model, that is, it did not appear in the training data. Specifically, the modifier of VA expressions in the test data appeared before the source, for example, in "Wir brauchen einen neuen Don Quijote.", "Der russische James Bond heißt Stierlitz.", or "Eine griechische Cathy Freeman zu haben wäre nicht schlecht."[8]. In these cases, the model did not tag any word as part of a VA chunk but it should have tagged 'new', 'Greek' and 'Russian' as modifiers, 'Don Quijote', 'James Bond' and 'Cathy Freeman' as source, and 'Stierlitz' as target in the second sentence, whereas the other sentences do not have a target. This is a limitation of the model, even though in some other of these specific expressions, it tagged the chunks correctly.

6 Discussion

We analyzed cross-lingual VA extraction using English as source language and German as target language with limited annotated data. Our models achieve strong results which are even comparable to monolingual approaches like Schwab et al. (2022), also in the robustness study. Translating the training data to the target language worked best.

One goal for the future is to make more use of the semantics, even though this is a whole new problem as the examples have to be analyzed much deeper. Also, the generation of VA is a task we want to follow.

[6] "Mike Stern is the roughneck of jazz guitarists."

[7] "AIDS – Worst-case scenario of Genetic Engineering?"

[8] "We need a new Don Quijote.", "The Russian James Bond is called Stierlitz.", "Having a Greek Cathy Freeman would not be bad."

References

1990. TRAVEL ADVISORY. *The New York Times*.

2005. Taz-archiv.

2013. Unter Wetterfröschen. *Der Spiegel*.

Ehsan Aghazadeh, Mohsen Fayyaz, and Yadollah Yaghoobzadeh. 2022. Metaphors in pre-trained language models: Probing and generalization across datasets and languages. *arXiv preprint arXiv:2203.14139*.

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.

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

Frank Fischer and Robert Jäschke. 2019. 'The Michael Jordan of greatness'—Extracting Vossian antonomasia from two decades of The New York Times, 1987–2007. *Digital Scholarship in the Humanities*, 35.

Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online. Association for Computational Linguistics.

Robert Jäschke, Jannik Strötgen, Elena Krotova, and Frank Fischer. 2017. "Der Helmut Kohl unter den Brotaufstrichen". Zur Extraktion Vossianischer Antonomasien aus großen Zeitungskorpora. In *Proceedings of the DHd 2017*, DHd '17, pages 120–124. Digital Humanities im deutschsprachigen Raum.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319, 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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Sam Roberts. 1992. METRO MATTERS; Spirit of Newburgh Past Haunts Political Present. *The New York Times*.

Evan Sandhaus. 2008. The New York Times Annotated Corpus LDC2008T19. DVD, Linguistic Data Consortium, Philadelphia.

Michel Schwab, Robert Jäschke, Frank Fischer, and Jannik Strötgen. 2019. "A Buster Keaton of Linguistics": First automated approaches for the extraction of vossian antonomasia. 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 6238–6243, Hong Kong, China. Association for Computational Linguistics.

Michel Schwab, Robert Jäschke, and Frank Fischer. 2022. "The Rodney Dangerfield of Stylistic Devices": End-to-end detection and extraction of Vossian Antonomasia using neural networks. *Frontiers in Artificial Intelligence*. (in print).

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 248–258, Baltimore, Maryland. Association for Computational Linguistics.

Martin Víta. 2020. Cross-lingual metaphor paraphrase detection – experimental corpus and baselines. In *Information and Software Technologies*, pages 345–356, Cham. Springer International Publishing.

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.

The Elementary Scenario Component Metric for Summarization Evaluation

Martin Kirilov[1, 2], **Daan Kolkman**[1, 2, 3], **Bert-Jan Butijn**[2, 4]

[1]Eindhoven University of Technology, Eindhoven, The Netherlands
[2]Jheronimus Academy of Data Science, 's-Hertogenbosch, The Netherlands
[3]University of Applied Sciences Utrecht, Utrecht, The Netherlands
[4]Erasmus School of Accounting and Assurance, Erasmus University, Rotterdam, The Netherlands
marti.kkirilov@gmail.com, d.kolkman@tue.nl, bjbutijn@gmail.com

Abstract

Evaluation is a fundamental step in the development of novel automatic summarization methods. The correlation between commonly used automatic evaluation metrics and golden standard human evaluations is often modest at best. Automatic evaluation metrics have thus not proven an alternative to human evaluation. This presents a problem to the progress of automatic summarization because evaluations conducted by people are time-consuming, inconsistent, and costly. We introduce the Elementary Scenario Component Metric (ESCM), which draws on the creative arts and scenario modelling literature. This metric does not require reference summaries, but uses twelve elementary scenario components, or a sub-selection thereof, to estimate the relevance of summaries instead. We show that the ESCM achieves a correlation of 0.89 with human evaluations and is less time-consuming than the creation of reference summaries.

1 Introduction

Although automatic summarization has a long history (Luhn, 1958), it remains a key challenge within Natural Language Processing (Fabbri et al., 2019). The aim of automatic summarization is to shorten a source text into a condensed version, conserving both the information content and the overall meaning (Kiyani and Tas, 2017). Automatic summarization methods can be classified among two axes: the summarization method and the number of input texts.

Irrespective of the automatic summarization method, an essential step in the development of a summarization system is the evaluation of generated summaries. Evaluation, however, is not without issues. Evaluation protocols differ from one paper to the next (Hardy et al., 2019) and evaluation metrics such as ROUGE are often used well beyond their intended scope (Liu and Liu, 2008). Moreover, (Fabbri et al., 2021) demonstrate that the system-level correlations between fourteen commonly used evaluation metrics and golden standard human evaluations for coherence, consistency, fluency, and relevance are mostly weak to moderate.

A commonality among most evaluation techniques is the need for reference summaries, referred to as gold-standard summaries. Most evaluation techniques calculate a score based on the comparison of the system generated summaries with the reference summaries. A drawback of employing reference summaries is that objectively establishing them is difficult (Steinberger and Ježek, 2009b). These reference summaries are human-written and therefore introduce a considerable level of subjectivity, since there is not a single perfect way of writing a text summary (Saziyabegum and Sajja, 2016). Moreover, writing these reference summaries by humans can be time-consuming and costly for large corpora (Giannakopoulos and Karkaletsis, 2013).

This paper presents the Elementary Scenario Component Metric (ESCM) which is grounded in work on scenarios in the creative arts and scenario modelling literature (De Kock, 2014). The ESCM does not require reference summaries, but utilizes elementary scenario components to estimate the relevance of summaries instead. The contribution of this study to the automatic summarization literature is twofold: ESCM reduces the dependence on people as human-written reference summaries are no longer a requisite for the evaluation of automatic summarization methods. More importantly, the ESCM is grounded in the creative arts literature and has a correlation of 0.89 with human evaluations, suggesting it may be a better proxy than other metrics currently in use. This paper is structured as follows: First we provide a brief overview of the literature on evaluation metrics for automatic summarization. Next we discuss the concept of scenarios as used in the creative arts and scenario planning literature. This informs discussion of the twelve Elementary Scenario Components. We then

introduce the ESCM and apply a sub-selection of five elementary scenario components in an experiment of multi-document crime case summarization. We conclude by offering some reflections and limitations of our work and offer avenues for further development of the ESCM. Our code can be found in the paper's GitHub page[1].

2 Related work

2.1 Evaluation protocols

Although much progress has been made, there is no consensus on how automatic summarization systems should be evaluated (van der Lee et al., 2019). A variety of metrics and procedures exist, Steinberger and Ježek (2009a) put forward a taxonomy of automatic summarization evaluation techniques. They suggest evaluation techniques can be broadly classified in two categories: intrinsic and extrinsic.

2.1.1 Intrinsic Evaluation

Intrinsic methods are based on how well the summary information content matches the information of a reference summary (Murray et al., 2008). Intrinsic evaluation can be further broken down into text quality evaluation and content evaluation. Evaluating the quality of the text is usually done by people, who rate different aspects of the summary on a predefined scale. These aspects of linguistic quality include grammatically, non-redundancy, reference clarity, and coherence and structure (Steinberger and Ježek, 2009a).

Content evaluation consists of co-selection measures such as precision, recall, and F-score, which ignore the fact that sentences can contain the same information even if written different and content-based measures which do not have that limitation. Content-based measures compare the words in a sentence, rather than the entire sentence, examples include cosine similarity (Louis and Nenkova, 2008), longest common subsequence, n-gram matching, pyramids (Nenkova and Passonneau, 2004), and Latent Semantic Analysis (LSA) based measures (Steinberger and Ježek, 2009a). The disadvantage of such measures is that they do not discriminate very well between summaries that involve differences in meaning (Mani, 2001). In effect, these measures are likely to work with extractive systems better than abstractive ones (Aries et al., 2019).

[1] https://github.com/ESCM-summarization/ESCM-evaluation

The most notable example of n-gram matching is the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric introduced by Lin (2004) which has been the de-facto standard for automatic evaluation of summarization in recent years (Yao et al., 2017). It works by measuring similarity between system generated summaries and reference summaries. Depending on the implementation, it can measure the overlap of uni-grams (ROUGE-1), bi-grams (ROUGE-2), Longest Common Subsequence (ROUGE-L), and others.

Aside from ROUGE, there are other metrics that have been used for summary evaluation like BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and BLANC (Lita et al., 2005). However, unlike ROUGE, these metrics were originally developed for evaluation of machine translation systems. Consequently, the use of these metrics in the automatic summarization literature is very limited.

A key problem with intrinsic evaluation is that these methods need to match the result summary with an "ideal summary", which is difficult to establish for a number of reasons (Steinberger and Ježek, 2009a). When people have to pick the most relevant sentences from documents in order to produce summaries, they frequently disagree in which sentences best represent the content of a document (Spärck-Jones et al., 2007). There is thus an inherent subjectivity to summarization evaluation (Lloret et al., 2018). Moreover, manual evaluation is expensive and the obtained results may be difficult to reproduce (Giannakopoulos and Karkaletsis, 2013).

More recently, researchers have developed various novel evaluation approaches which do improve upon the well-established intrinsic evaluation methods. For instance BERTScore (Zhang* et al., 2020), originally aimed at other tasks such as machine translation and image captioning. Or QAEval (Deutsch et al., 2021) and QuestEval (Scialom et al., 2021), which are both based on question-answering (QA) approaches. The latter might even be comparable to the ESCM, since it also doesn't require any ground-truth reference. Nonetheless, all three of these metrics require the use of additional models only for the evaluation of summaries. This, undoubtedly, would introduce a lot more unknowns within a summarization pipeline, and make it significantly more complex.

2.1.2 Extrinsic Evaluation

Extrinsic evaluation techniques determine the quality of a summary based on how it affects other tasks - a summary is considered good if it helps with solving these tasks (Gambhir and Gupta, 2017). These techniques are also known as task-based methods. Extrinsic evaluations have the advantage of assessing the utility of summarization in a task, so they can be of tremendous practical value to users of summarization technology (Mani, 2001). On the other hand, they are less helpful in providing insights on how the system actually performs the summarization. According to (Steinberger and Ježek, 2009a), the three most relevant tasks for extrinsic evaluation are document categorization, information retrieval, and question answering.

In the case of document categorization, the evaluation seeks to determine whether the generic summary is effective in capturing whatever information in the document is needed to correctly categorize the document (Steinberger and Ježek, 2009a). A document corpus and the topics of each document are needed in order to apply this method. The results of categorizing summaries are compared to results of categorizing full texts and random sentence extracts (Steinberger and Ježek, 2009a). The main metrics used in this case are the precision and recall of the categorization (or also their F-1 score) (Steinberger and Ježek, 2009a).

In the context of Information Retrieval (IR), summaries and full documents are used as input to an IR system. The similarity between how well the system works with the summaries as opposed to the full documents should serve as an indicator of the quality of summaries (Steinberger and Ježek, 2009a). Steinberger and Ježek suggest several methods to measure this similarity, namely Kendall's tau, Spearman's rank correlation, and linear correlation.

Question-answering is the third relevant task suggested by (Steinberger and Ježek, 2009a). The task is generally about reading comprehension — a human reads original documents or summaries and then answers a multiple-choice test (Mani, 2001). The idea is that if reading a summary allows a human to answer questions as accurately as they would by reading the original document, the summary is highly informative (Mani, 2001).

2.2 Elementary Scenario Components

Intrinsic evaluation techniques have a simple idea in common, they all compare two elements: the subject of evaluation (a text summary) and some kind of reference object (a reference summary). Scholars often use automatic summarization evaluation techniques that need either reference summaries, or some kind of specific task in order to measure the quality of a system generated summary.

With the Elementary Scenario Component Metric (ESCM), we employ a different approach that does not require reference summaries. Instead, our method is based on the idea that all relevant aspects of a some narrative can be described by the means of twelve elementary scenario components (De Kock, 2014). In the creative arts literature, a narrative is generated by a scenario that describes the interactions between characters (ibid.).

There is a rich literature on the nature and role of scenarios throughout history which can be traced back to Artisotle's Poetics. Aristotle is credited for being the first to distinguish between different components of scenarios and many have followed in his footsteps (Janko et al., 1984). Based on an extensive review of the literature on scenarios components in the creative arts, De Kock (2014) identified twelve Elementary Scenario Components (ESC-12) as the building blocks for any scenario. A list of all the components with a description is provided in Table 1. In this paper we suggest that these elementary scenario components can provide the foundation for a new automatic summarization system evaluation metric.

Our work is most closely related to intrinsic evaluation techniques based on "factoids". Such techniques attribute a score to text fragments based on their informativeness. These fragments are considered single coherent semantic units, such as "the Netherlands", "glass of water", and "the car arrived" (Van Halteren and Teufel, 2003). Radev et al. (2004) emphasized that it is necessary to determine not only what factoids should be included in the summary, but also how important they are. The pyramid method introduced by Nenkova and Passonneau (2004) builds on this idea to leverage multiple manually generated summaries. It demonstrates that factoids can be assigned weights based on those references summaries and how highly weighted units can be considered as more essential for a summary than not so highly weighted ones.

Component	Type	Description
Arena	Objective	The location where the story takes place.
Time(frame)	Objective	The time(frame) in which the story takes place.
Context	Objective	The set of circumstances that surround the story.
Protagonist	Objective	The main character of the story around whom the plot evolves.
Antagonist	Objective	The opposition against whom/which the protagonist must contend.
Motivation	Subjective	The psychological features that drive the protagonist.
Primary objective	Subjective	The way by which the protagonist attains his motivation.
Means	Objective	The methods or instruments by which the protagonist achieves his primary objective.
Modus operandi	Objective	The method of operation of the protagonist.
Resistance	Objective	The obstacles the protagonist has to overcome to be able to achieve his objective.
Symbolism	Interpretable	When a component carries a symbolic value for the protagonist, antagonist, or the audience.
Red herring	Interpretable	A misleading occurrence or indicator used to lead someone in the wrong direction of thought.

Table 1: A list of the ESC12 (De Kock, 2014)

In terms of these contributions, the ESCM provides a more general framework for determining relevant factoids, which could then be ranked using the pyramid method.

The ESC-12 are divided into three categories - objective, subjective, and interpretable components: Objective components comprise observable phenomena and are not related to the protagonist's individual feelings and interpretations, Subjective components reflect the protagonist's individual interpretation of experiences and interpretable components do not have a meaning until interpreted by a third party (De Kock, 2014).

3 Elementary Scenario Components Metric

We propose a new evaluation metric based on the ESC-12. Below we introduce the procedure for applying the ESC-12 to an automatic summarization system.

1. Determine relevant ESC and operationalize variable mapping. De Kock (2014) argues that ESC-12 represent general categories that occur in any type of scenario. However, careful tuning of the variables making up the components is necessary to fit a particular domain. For instance, although the *Arena* may be relevant to a narrative on historic geography and a narrative pertaining a criminal case, their operationalization would be different.

2. Annotate data. The metric requires an annotated dataset, with labels for (sub)set of ESCs-12 for each article or story (one or more documents can be referring to one story). After a system generates summaries for the input dataset, the evaluation metric is calculated based on the presence of the corresponding ESCs for each story in each summary relative to the presence of these components in the input texts. The formula which is used to calculate this can vary depending on fine-tuning for optimal results.

3. Compute ESCM. In the third step, the ESCM is computed. This requires ESC labeled texts, and summaries generated on these texts by the summarization system.

3.1 Elementary Scenario Component Metric Calculation

The first step is to determine which of the labelled components are available in the pre-processed input texts. This is of importance, because some components might be missing from the input texts after certain levels of pre-processing. For instance, if the inputs are truncated to 500 tokens, some components might not be included. Another possibility - for multi-document summarization- is that a component is present in one source text, which has been discarded in the selection step (e.g. if the Protagonist is mentioned only in document #3, but documents #1 and #2 are selected for truncation).

The second step consists of determining which of the components in the input texts are present in the summaries. This is a fairly complicated process in the context of automatic summarization, therefore we illustrate it with an example. Suppose that the algorithm is trying to match the name of the antagonist obtained from the dataset and present in the input text (e.g. John Doe). In case the character string "John Doe" is also in the summary, there is a 100% match. The problem arises when the summary contains some variant of the antagonist's name, for instance "John D." or "J. D.". Therefore, we include an approximate string matching technique in the evaluation metric algorithm. This allows the algorithm to distinguish between complete and partial string matching.

In the final step, the algorithm calculates a sum of the individual scores of the components of inter-

est. Each individual score can vary between 0 and 100, where 100 means that the two components are identical, and the smaller the score gets, the more different the components are. Equation 1 illustrates how the evaluation score is calculated:

$$\text{ESCM} = \frac{\sum_{(M,R)\in C} \frac{\sum_{\substack{x\in M \\ y\in R}} \frac{FuzzyScore(x,M)}{FuzzyScore(y,R)}}{K}}{N} \quad (1)$$

Where N is the total number of cases, M and R are a pair of model generated summary and reference input text for a single case from the set of all cases C. Furthermore, x and y are pairs of ESCs from the model generated summary M and the reference input text R respectively. K is the number of components that are considered when calculating the metric, which in this case is 5. $FuzzyScore(arg_1, arg_2)$ is a function that returns a number between 0 and 100 depending on how accurately is the ESC label arg_1 represented in the text arg_2.

Finally, a specific threshold is introduced for each of the components. The idea behind these thresholds is to determine if the matched component is referring to the same n-gram as the label, or it is a completely different n-gram. Again, an example may help illustrate how this works: suppose the Protagonist's name is "George S.", but the best match the algorithm is able to find is the string "was reported" with a score of 60. However, if the Protagonist's name was "Brian Nijhof", then the best match would be "Brian N." with a score of 74. The introduction of thresholds helps the algorithm to discard the first example, but keep the second.

The thresholds are used as follows: For each case the algorithm checks if all of the ESCs have a score above the threshold in the input texts for that case. If any of the ESCs have a lower score, the whole case is discarded from the calculation of the evaluation metric. This is necessary to ensure that all cases used in the calculation have all ESCs in the input texts. However, the thresholds are used in a different way when handling the summary texts. For each case, if the summary ESC score is below the threshold it means that it is wrong, and is thus set to 0. This process is illustrated with a few examples in Table 2.

In the first example, the name of the protagonist is mentioned with a score of 70 in the input texts, which is just on the threshold, and consequently, it is also checked in the summary text. However, in the summary, the fuzzy matching returns a score of 46, which is below the threshold. Then the algorithm sets the ESC summary score to 0, and the ESC relative score becomes $0/70 = 0\%$. The second example shows that if the fuzzy score of an ESC is below the threshold for the input texts, the ESC does not receive a relative score, and the whole case is discarded. For the last two examples, both of the input texts and summary text fuzzy scores are above the thresholds, and therefore the relative score can be calculated as a fraction between the fuzzy scores.

3.2 Fuzzy String Matching

For multi-document summarization there is higher chance of discrepancies between the data labels and the actual strings these labels refer to in the articles. Information in some news articles might be missing or different compared to other news articles. For example, let us say "John Doe" is the antagonist label for a case, while some of the articles only contain the name "John D.". Consequently, it would be required to measure how similar these two strings are. Directly matching strings is not a viable option, because there are many examples where the difference is only a few characters, and it is evident that they are referring to the same thing. Thus, there is a need for a method which would allow for the implementation of fuzzy string matching. Such a method is Levenshtein distance (Levenshtein, 1966), of which we used the Token Sort Ratio method for our FuzzyScore.

4 Experiment

To demonstrate the effectiveness of the ESCM, we conduct an experiment with automatic summarization of crime cases. There are plenty examples in literature for application of NLP techniques for crime data analysis (Ku and Leroy, 2014; van Banerveld et al., 2014; Ku et al., 2008; Iriberri and Leroy, 2007; Wang et al., 2007). The vast majority of such studies focus on information extraction (e.g. Named-entity recognition), crime classification, and crime analysis (Ku and Leroy, 2014). However, there is a lack of research about automatic summarization in the crime domain. The only example of a summarization system for crime texts that was found at the time of writing is the SALOMON project (Moens et al., 1997; Moens, 2000). For our experiment we implement three models: Hi-Map (Fabbri et al., 2019), Transformer

ESC label	Input texts		Summary text		ESC Relative score
	ESC match	Fuzzy score	ESC match	Fuzzy score	
Rudolf Käsenbier	Rudolf K.	70	of Enschede	46	0%
Anouar B.	book and	50	-	-	-
Henk Haalboom	Henk Haalboom	100	Haalboom	76	76%
Michael E.	Michel E.	94	Michel E.	94	100%

Table 2: Examples of different combinations of fuzzy scores and their corresponding relative score for the Protagonist ESC in the Homicide dataset.

(Vaswani et al., 2017), and TextRank (Mihalcea, 2004). For the first two models we tried a truncation of 500 and 1000 tokens. For TextRank we only implemented a 500 token model. These truncation lengths are chosen to be inline with the setting of relevant studies such as Fabbri et al. (2019)

4.1 Homicide dataset

The original version of the Homicide dataset has been created by Pandora Intelligence. It consists of 100 manually chosen homicide cases that occurred in the Netherlands. For each case there are relevant data about some of the ESCs (usually not all), as well as multiple source articles about the case. These articles were web-scraped from manually selected URLs. The methodology of selecting these URLs is mainly based on the results of Google Search queries for the most popular Dutch homicide cases from the last few decades. The dataset comes in a Dutch and English version. The Dutch version is created from web-scraping these URLs of mostly Dutch news websites. The English version is made by domain experts, who automatically translated the Dutch version via Google Translate API[2], and manually reviewed all translations.

On average, the number of articles per case is 13.86 and there are no cases with less than five sources. This high number of source articles makes the Homicide dataset very suitable for multi-document summarization. A more detailed information about the distribution of the source articles can be found in Table 4.

4.2 Application of the ESCM procedure

The ECSM procedure offers an effective and durable set of components to describe, characterise and model a criminal incident (De Kock, 2014). We follow the procedure outlined in section three. For means of illustration, and to limit the degree of subjectivity in the labelling, we exclude both subjective and interpretable components. In *Step 1* of the procedure, we thus select the following objective components: *Arena*, *Timeframe*, *Protagonist*, *Antagonist*, and *Modus operandi*.

Next, we provide the operationalization of these concepts in the context of our experiment. More specifically, we specify the variable or set of variables for each ESC that we have selected. Table 3 provides an overview of this mapping. A subset of the Homicide dataset was then made, only consisting of cases, which contain annotations about all 5 pre-selected objective components mentioned in *Step 2*. As a result, this yielded 26 cases which were suitable for the application of our ESCM evaluation. In *Step 3* we compute the ESCM using the following thresholds for fuzzy string matching: Arena 65, Timeframe 75, Protagonist 70, Antagonist 75, Modus operandi 75. The thresholds were manually fine-tuned upon basic data exploration and characteristics of the ESC annotations.

4.3 Evaluation

We evaluate our procedure and the ESCM using a questionnaire administered to several expert respondents. We followed the guidelines for expert evaluation proposed by van der Lee et al. (2019). Although evaluation by a more general audience is sometimes preferred, we opted for expert evaluation in an effort to collect the highest quality data.

The survey for the human evaluation experiment was distributed among police offcers from the Dienst Regionale Informatieorganisatie (DRIO) department of Police Oost-Brabant. In total, 21 participants filled in the survey. This can be regarded as a high number of participants for an expert-focused study, which typically use up to four experts (van der Lee et al., 2019). The cases presented in each variant were randomly selected from the 26 available.

4.3.1 Text quality criteria

It is very common for automatic summarization studies which perform human evaluation to report text quality. However, text quality criteria differ across tasks, and there is a significant variety in

[2] https://cloud.google.com/translate

Component	Operationalization
Arena	The city where the homicide took place.
Timeframe	The date on which the crime took place, which is provided in a DD-MM-YYYY format.
Protagonist	The name of the murderer or murderers.
Antagonist	The name of the victim or victims.
Means	The murder weapon.
Modus operandi	The type of murder (e.g. manslaughter or first degree murder).

Table 3: The operationalization of the ESCM for the Homicide dataset

Number of sources	Frequency
Up to 5	2
From 6 to 10	23
From 11 to 15	44
From 16 to 20	23
More than 20	8

Table 4: Distribution of source articles in the Homicide dataset.

naming conventions for measures of text quality (van der Lee et al., 2019). There is also absence of common evaluation guidelines for NLG tasks (Belz and Hastie, 2014), which means that the measured criteria should be explicitly defined when implementing a human evaluation experiment (van der Lee et al., 2019). Although the ESCM is primarily intended to measure accuracy, following van der Lee et al. (2019) also measure relevance, and fluency as text quality criteria.

4.3.2 Questionnaire design

The survey consisted of three parts. Participants were presented with an introduction to the research topic, the ESC framework, and the goals of the survey. The second part was comprised of five different text summaries, each followed by a set of ESC-related questions and two text quality questions. The third and final part included general demographic questions and concluded the survey.

Based on the goals of this evaluation experiment, it was decided to include three types of questions – text quality questions, questions evaluating the accuracy of ESCs in various text summaries, and general demographic questions. For the first two types, we use a 5-point Likert scale, the full survey can be found on GitHub.

4.4 Results

4.4.1 Human evaluation

The average scores for each case-summary combination are reported in Table 5. Let X be the variable containing ESCM scores of multiple case summary variants. More specifically, X contains the 15 scores labelled as *ESCM* in Table 5. Let Y be the variable containing the average scores per case and summary variant obtained from the survey results. Y contains the 15 scores labelled with *Survey* in Table 5. Upon calculating Pearson's Correlation Coefficient, the variables X and Y were found to be strongly positively correlated $(\boldsymbol{r(13) = .89, p < .001})$. A scatter plot with a trend line is illustrated in Figure 1.

The results of the text quality, averaged per criterion are presented in Table 6. As with the first part of the questionnaire, we used a 5-point Likert scale. Expectedly, the Transformer models score best in terms of subjective fluency and relevance, even though the results for all model combinations are quite close to the average on the scale.

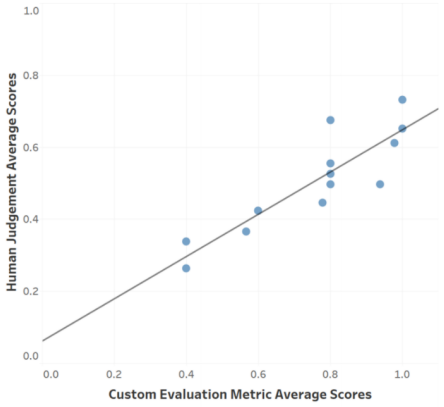


Figure 1: Scatter plot of the average survey and ESCM scores.

van der Lee et al. do recommend to always report inter-rater reliability (IRR). IRR measures degree of consensus among ratings provided by various human evaluators. Krippendorff's alpha Krippendorff (1970) is a frequently used IRR measure and can be used regardless of the number of observers, levels of measurement, sample sizes, and presence or absence of missing data Krippendorff (2004). Three distinctive groups of summaries were presented to three different groups of participants, we report Krippendorff's alpha for each in Table 7. The coefficients show positive agreement among the participants in the human evaluation.

Case ID	Score type	Summary variant (model, truncation, selection)				
		Hi-MAP 500; 3	Transformer 500; 3	TextRank 500; 3	Hi-MAP 1000; 2	Transformer 1000; 2
Case 44	ESCM	77.80	100.00	80.00	40.00	97.80
	Survey	64.57	85.14	87.43	53.71	81.14
Case 12	ESCM	80.00	60.00	100.00	93.80	56.60
	Survey	69.71	62.29	93.14	69.71	56.57
Case 31	ESCM	40.00	80.00	80.00	80.00	40.00
	Survey	53.71	75.43	72.57	75.43	46.29

Table 5: Average scores of the ESCM and the survey results.

Criterion	Summary variant (model, truncation, selection)				
	Hi-MAP 500; 3	Transformer 500; 3	TextRank 500; 3	Hi-MAP 1000; 2	Transformer 1000; 2
Fluency	2.52	**3.14**	2.67	2.67	3.05
Relevance	2.57	**3.14**	3.10	2.86	2.29

Table 6: Text quality criteria average scores for each of the summary variants presented in the human evaluation experiment.

Case ID	Number of annotators	Number of questions	Krippendorff's alpha
Case 44	7	25	.38
Case 12	7	25	.60
Case 31	7	25	.36

Table 7: Inter-Rater Reliability coefficients for the human evaluation experiment.

4.4.2 Robustness check

To further explore the robustness of our findings we asked independent annotators to write golden standard summaries for the same homicide cases used in our previous experiments, namely 12, 31, and 44. We calculated four different ROUGE measures per case and averaged all ROUGE F-1 scores for all cases per ROUGE type. We calculated the Pearson's Correlation Coefficient of these values in relation to the average ESCM and Human evaluation scores. The results are presented in Table 8. The correlation of the best match-up (R-SU4 and human evaluators) is considerably lower than that of the ESCM and the human evaluation scores. R-SU4 performs best out of the ROUGE metrics which in line with early findings (Lin, 2004) and thus is further proof of the robustness of our experiment.

	R-1	R-2	R-L	R-SU4
ESCM	0.60	0.49	0.53	0.67
Human	0.64	0.58	0.63	0.72

Table 8: Correlation coefficients for the ESCM, human evaluation scores, and various ROUGE scores.

5 Conclusion

We presented a novel automatic evaluation metric for automatic summarization based on the ESC framework: the ESCM. The ESCM is a recall-based metric and we recommend it be used alongside precision-based metrics. With our experiments, we demonstrate the capabilities of the ESCM. However, our metric is subject to some limitations. With only 26 cases, the dataset we used is relatively small. Furthermore, in relation to the calculation of the ESCM, we used a threshold to determine if an ESC is contained in a text or not. These thresholds were determined based on the results from the fuzzy string matching. It is unclear whether these thresholds would be generalizable to other (non)homicide-related datasets.

Our findings show the potential of the ESCM, but more research is necessary to explore its usefulness. Future work could experiment with the ESCs by including more components from the subjective or interpretable type. Furthermore, the selected components could be utilized better by increasing the level of detail (e.g. using the full date instead of just the year). A next step in the validation of the ESCM could be done using the dataset from Text Analysis Conference (TAC) 2010 summarization track. Part of the TAC 2010 dataset consists of texts and summaries of criminal attacks, that are labelled in a similar manner to the ESCM.

Although the primary focus of the ESCM is relevance, it achieved a strong correlation with the average of human evaluations for relevance, accuracy, and fluency. Moreover, those that labelled the texts and wrote summaries reported that labelling for the ESCs was considerably less time consuming. Although this is merely anecdotal evidence, it echoes findings by Lloret et al. (2013) who suggest that producing reference summaries takes 8 – 10 times longer than answering a series of questions about a text. Based on this and the high correlation with human evaluation, we believe the ESCM may present a useful alternative to existing metrics, especially in applications domains that are under-resourced or where writing and evaluating summaries requires domain expertise.

References

Abdelkrime Aries, Walid Khaled Hidouci, et al. 2019. Automatic text summarization: What has been done and what has to be done. *arXiv preprint arXiv:1904.00688*.

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, Ann Arbor, Michigan. Association for Computational Linguistics.

Anja Belz and Helen Hastie. 2014. *Comparative evaluation and shared tasks for NLG in interactive systems*, page 302–350. Cambridge University Press.

P. De Kock. 2014. *Anticipating criminal behaviour: Using the narrative in crime-related data*. Ph.D. thesis, Tilburg University.

Daniel Deutsch, Tania Bedrax-Weiss, and Dan Roth. 2021. Towards question-answering as an automatic metric for evaluating the content quality of a summary. *Transactions of the Association for Computational Linguistics*, 9:774–789.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47:1–66.

George Giannakopoulos and Vangelis Karkaletsis. 2013. Summary evaluation: Together we stand npower-ed. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 436–450. Springer.

Hardy Hardy, Shashi Narayan, and Andreas Vlachos. 2019. HighRES: Highlight-based reference-less evaluation of summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3381–3392, Florence, Italy. Association for Computational Linguistics.

A. Iriberri and G. Leroy. 2007. Natural language processing and e-government: Extracting reusable crime report information. In *2007 IEEE International Conference on Information Reuse and Integration*, pages 221–226.

Richard Janko et al. 1984. *Aristotle on comedy: towards a reconstruction of Poetics II*, volume 2. Univ of California Press.

Farzad Kiyani and Oguzhan Tas. 2017. A survey automatic text summarization. *Pressacademia*, 5(1):205–213.

Klaus Krippendorff. 1970. Estimating the reliability, systematic error and random error of interval data. *Educational and Psychological Measurement*, 30(1):61–70.

Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology (second edition)*. Sage Publications.

Chih Hao Ku, Alicia Iriberri, and Gondy Leroy. 2008. Natural language processing and e-government: crime information extraction from heterogeneous data sources. In *Proceedings of the 9th Annual International Conference on Digital Government Research, Partnerships for Public Innovation, DG.O 2008, Montreal, Canada, May 18-21, 2008*, volume 289 of *ACM International Conference Proceeding Series*, pages 162–170. Digital Government Research Center.

Chih-Hao Ku and Gondy Leroy. 2014. A decision support system: Automated crime report analysis and classification for e-government. *Government Information Quarterly*, 31(4):534 – 544.

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

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.

Lucian Vlad Lita, Monica Rogati, and Alon Lavie. 2005. Blanc: Learning evaluation metrics for mt. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, page 740–747, USA. Association for Computational Linguistics.

Feifan Liu and Yang Liu. 2008. Correlation between ROUGE and human evaluation of extractive meeting summaries. In *Proceedings of ACL-08: HLT, Short Papers*, pages 201–204, Columbus, Ohio. Association for Computational Linguistics.

Elena Lloret, Laura Plaza, and Ahmet Aker. 2013. Analyzing the capabilities of crowdsourcing services for text summarization. *Language resources and evaluation*, 47(2):337–369.

Elena Lloret, Laura Plaza, and Ahmet Aker. 2018. The challenging task of summary evaluation: an overview. *Language Resources and Evaluation*, 52(1):101–148.

Annie Louis and Ani Nenkova. 2008. Automatic summary evaluation without human models. In *TAC*.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Inderjeet Mani. 2001. Summarization evaluation: An overview. In *Proceedings of the 2nd NTCIR Workshop on Research in Chinese and Japanese Text Retrieval and Text Summarization*, Tokyo: National Institute of Informatics.

Rada Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 170–173, Barcelona, Spain. Association for Computational Linguistics.

Marie-Francine Moens. 2000. *Automatic Indexing and Abstracting of Document Texts*, volume 6. Springer Science & Business Media.

Marie-Francine Moens, Caroline Uyttendaele, and Jos Dumortier. 1997. Abstracting of legal cases: The salomon experience. In *Proceedings of the 6th International Conference on Artificial Intelligence and Law*, ICAIL '97, page 114–122, New York, NY, USA. Association for Computing Machinery.

Gabriel Murray, Thomas Kleinbauer, Peter Poller, Steve Renals, Jonathan Kilgour, and Tilman Becker. 2008. Extrinsic summarization evaluation: A decision audit task. In *Machine Learning for Multimodal Interaction*, pages 349–361, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ani Nenkova and Rebecca J Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the human language technology conference of the north american chapter of the association for computational linguistics: Hltnaacl 2004*, pages 145–152.

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.

Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938.

Saiyed Saziyabegum and Priti S. Sajja. 2016. Literature review on extractive text summarization approaches. *International Journal of Computer Applications*, 156(12):28–36.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, Jacopo Staiano, Alex Wang, and Patrick Gallinari. 2021. QuestEval: Summarization asks for fact-based evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6594–6604, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Karen Spärck-Jones, Stephen E Robertson, and Mark Sanderson. 2007. Ambiguous requests: implications for retrieval tests, systems and theories. In *ACM SIGIR Forum*, volume 41, pages 8–17. ACM New York, NY, USA.

Josef Steinberger and Karel Ježek. 2009a. Evaluation measures for text summarization. *Computing and Informatics*, 28(2):251–275.

Josef Steinberger and Karel Ježek. 2009b. Text summarization: An old challenge and new approaches. In *Foundations of Computational, IntelligenceVolume 6*, pages 127–149. Springer.

Maarten van Banerveld, Nhien An Le-Khac, and M. Tahar Kechadi. 2014. Performance evaluation of a natural language processing approach applied in white collar crime investigation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8860:29–43.

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.

Hans Van Halteren and Simone Teufel. 2003. Examining the consensus between human summaries: initial experiments with factoid analysis. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, pages 57–64.

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.

X. Wang, D. E. Brown, and J. H. Conklin. 2007. Crime incident association with consideration of narrative information. In *2007 IEEE Systems and Information Engineering Design Symposium*, pages 1–4.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2017. Recent advances in document summarization. *Knowledge and Information Systems*, 53(2):297–336.

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*.

Scaling Native Language Identification with Transformer Adapters

Ahmet Yavuz Uluslu
Universität Zürich & PRODAFT
ahmetyavuz.uluslu@uzh.ch

Gerold Schneider
Universität Zürich
gschneid@cl.uzh.ch

Abstract

Native language identification (NLI) is the task of automatically identifying the native language (L1) of an individual based on their language production in a learned language. It is useful for a variety of purposes including marketing, security and educational applications. NLI is usually framed as a multi-class classification task, where numerous designed features are combined to achieve state-of-the-art results. Recently deep generative approach based on transformer decoders (GPT-2) outperformed its counterparts and achieved the best results on the NLI benchmark datasets. We investigate this approach to determine the practical implications compared to traditional state-of-the-art NLI systems. We introduce transformer adapters to address memory limitations and improve training/inference speed to scale NLI applications for production.

1 Introduction

Native Language Identification (NLI) is the task of automatically identifying the native language (L1) of an individual based on their writing or speech in another language (L2). It is used for a variety of purposes including marketing, security and educational applications. The growing interest in NLI from various research fields can be partly attributed to the outstanding performance of automated NLI systems against human annotators. A study of human performance in NLI (Malmasi et al., 2015) showed that NLI systems perform in the 80%–90% accuracy range while humans achieved 37.3% average accuracy. The experimentation also constrained the number of considered languages and texts to enable human competition.

NLI is most commonly framed as a multi-class classification problem. For text-based NLI, features are extracted from written resources produced by non-native speakers to train a classification model. The underlying hypothesis is that the L1 influences learners' second language writing as a result of the language transfer effect (Odlin, 1989). A variety of feature types have been explored to capture distinct features of the language interference phenomenon: spelling errors (Koppel et al., 2005; Chen et al., 2017); word and lemma n-grams (Tetreault et al., 2013); character n-grams (Kulmizev et al., 2017), dependency parsing and morphosyntax (Cimino et al., 2013). As seen by the two shared tasks on the NLI task organized in 2013 and 2017, the combination of such features produces the best outcomes for NLI (Tetreault et al., 2013; Malmasi et al., 2017). The top ranked systems made use of Support Vector Machine (SVM) models trained on a diverse set of linguistic features that can capture word, sentence and document level characteristics. (Markov et al., 2017; Cimino and Dell'Orletta, 2017)

Deep neural network based approaches were also considered for the NLI task. Bjerva et al. (2017) experimented with deep residual networks (DNN), long short-term memory networks (LSTM) and continuous bag-of-words embeddings to create meta classifiers. Li and Zou (2017) built an ensemble of single-feature SVMs fed into a multi-layer perceptron (MLP). Habic et al. (2020) integrated multi-task learning into convolutional neural networks (CNN) to create shared representations from multiple datasets. The studies concluded that traditional methods, i.e., SVM with engineered features, appear to work better than deep learning-based standalone and meta-classification approaches. Recently Lotfi et al. (2020) introduced the deep generative modelling approach to NLI which consists of fine-tuning a GPT-2 model to identify each language. Their method outperforms traditional machine learning approaches and currently achieves the best results on the benchmark NLI datasets.

The contributions of the work presented here are the following: (i) We investigate the resource requirements and inference performance for the deep generative approach in comparison to traditional state-of-the-art NLI systems, and (ii) we introduce ProDAPT, transformer adapters based on deep generative model to optimize memory and storage space, and (iii) we evaluate our approach on the NLI task and explore the tradeoffs.

2 Related Work

Deep Generative Approach. OpenAI's Generative Pre-trained Transformer-2 (GPT-2) is a unidirectional transformer-based language model pretrained on 40 GB of text data with the objective of predicting the next word given the context (Radford et al., 2019). It can generate coherent paragraphs of text and achieves state-of-the-art performance on many language modelling benchmarks without any task-specific training. Instead of training a classifier, the deep generative approach finetunes a generative model (GPT-2) on texts written by native speakers of each language (L1) to capture peculiarities of language transfer (Lotfi et al., 2020). After training N (number of target languages) models to learn the characteristics of each L1, they can be used to discriminate between unseen text samples based on the language model (LM) loss. The least LM loss is expected from the model that is trained on the same class (L1). Although there are examples of the LM loss as a ranking feature for other tasks such as substitute selection for text simplification (Uluslu, 2022), the deep generative approach is the first method to use such value as the only discriminator for text classification.

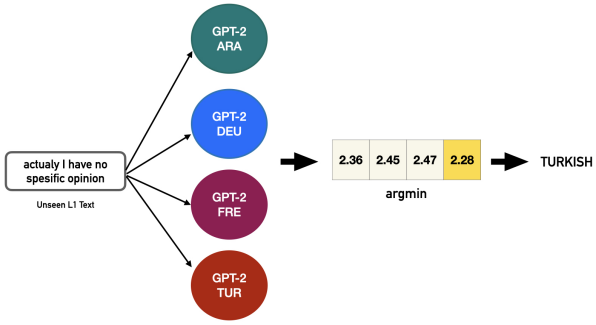


Figure 1: An example inference of an unseen text written by a Turkish native speaker.

Transformer Adapters. Adapters have been introduced as an alternative lightweight fine-tuning strategy that achieves equal performance to full fine-tuning on most tasks (Houlsby et al., 2019). They consist of a small set of additional newly initialized weights at every layer of the transformer. While the rest of the pretrained parameters of the large model are kept frozen during the fine-tuning process, these new parameters are actively trained on the target task. Efficient parameter sharing between tasks is possible by training several task-specific and language-specific adapters for the same model, which can be exchanged and combined afterwards. The code base for different state-of-the-art adapter architectures was integrated into the transformers library and released under the name adapter-transformers (Pfeiffer et al., 2020a). Recently, their adapters implementation started to support generative and seq2seq models such as GPT-2 (Sterz et al., 2021).

3 Data

We evaluate our approach on the most commonly used dataset in NLI research: the ETS Corpus of Non-Native Written English (TOEFL11) (Blanchard et al., 2013). The dataset contains 1,100 essays in English written by native speakers (L1) of 11 different languages: Arabic (ARA), Chinese (CHI), French (FRE), German (GER), Hindi (HIN), Italian (ITA), Japanese (JPN), Korean (KOR), Spanish (SPA), Telugu (TEL), and Turkish (TUR). In total there are 12,100 essays with on average 348 tokens per essay. The essays were written in response to eight different writing prompts, all of which appear in all 11 L1 groups, by authors with low, medium, or high English proficiency. The dataset is considered a benchmark dataset for NLI and was used in two shared tasks on the NLI task (Tetreault et al., 2013; Malmasi et al., 2017).

4 Methodology

We investigate the resource requirements and inference speed for the deep generative approach in three different subsections: storage requirements, memory requirements and inference speed.

Storage Requirements. The full deep generative approach from Lotfi et al. (2020) finetunes 11 gpt2-medium models to cover every language in the TOEFL11 dataset. A fine-tuned gpt2-medium model requires 1.4 GB of storage space. The training process with the early stopping of three validations is expected to take up to 61.6 GB of storage space upon completion. The inference (test) stage

only requires the best-performing models for every language to be kept. Therefore, 15.4 GB of storage space in total is needed to fully store the system.

Memory Requirements. To reach the final inference (target native language), each model needs to calculate the LM loss value for the given input. To fully load the parameters of 11 models in memory, 16.6 GB of GPU memory is required.

Speed Constraints. The deep generative approach does not inherently support parallelism. Further configuration in multi-threading and memory management is needed to accommodate different models in the GPU. The final inference still needs to be calculated in CPU time (argmin) and requires all model calculations to be completed beforehand. The number of models loaded simultaneously is constrained by the GPU requirements. In case all models are not available simultaneously, the memory operations to load/unload language models are part of the inference process and directly interfere with the performance.

Adapter Configurations. Two different configurations have been proposed for transformer adapters (Houlsby et al., 2019; Pfeiffer et al., 2020b). Rücklé et al. (2020) investigated the efficiency of two adapter architectures at training and inference time and found that they achieved comparable performance. Sterz et al. (2021) compared the performance of two architectures for adapter-based GPT-2 models on the GLUE benchmark and reported on-par results on different tasks. We also experimented with both architectures and found that Houlsby et al. (2019) produced better results for standalone GPT-2 models trained on NLI data and decided to implement our full architecture based on this configuration.

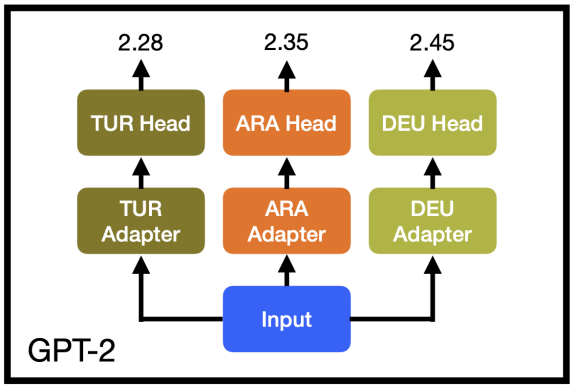


Figure 2: The ProDAPT architecture

The number of target languages for the NLI task can increase depending on the use case. In forensic linguistic investigations, particularly in the area of cybercrime, it may be necessary to cover up to 20-30 languages depending on the region and nature of crime, all of which may be in non-standardised forms, requiring the development of further models. To create a scalable NLI system, the memory bottleneck caused by the model size and the non-parallel nature of the deep generative approach should be addressed.

We implement ProDAPT architecture with transformer adapters to address these issues. We train an adapter for every L1 for 15 epochs with a learning rate of 1e-4. The original pretrained weights for the GPT-2 are kept intact, and the L1 (target language) information is compressed into the newly initialized parameters and the classification head. The storage space required for every language model (adapter + head) decreases to 218.7 MB. All adapters and their classification heads are loaded into a single gpt2-medium model and share the pretrained weights. To support parallel inference, the input is replicated at the first layer with L1 adapters. For every adapter, calculations are completed in parallel until the classification head is reached. The GPU memory required to load the ProDAPT architecture is 4.1 GB and the storage space requirement is 2.4 GB.

5 Results and Discussion

To compare the performance of our system with the state-of-the-art deep generative approach (Lotfi et al., 2020), we report the results in terms of classification accuracy on the TOEFL11 test set, as well as on the TOEFL11 dataset under 10-fold cross-validation (10FCV). We also report the storage and memory requirements to deploy our model in comparison to their approach. We use Titan T4 16 GB graphics card for the deep generative models and AMD EPYC 7702 64-Core CPU for Support Vector Machine (SVM) baselines where the GPU is not required. We think this is an acceptable choice since our work focuses on creating scalable NLI systems that can be easily deployed with widely available GPUs. We measure the inference speed with the time spent between CUDA events until the inference is finalized. We warm up the GPU before the test and repeat the experiment 100 times to enable robust results. Since the deep generative approach does not require feature engineering, we create a unigram SVM baseline to ensure a fair comparison. We compare the performance against

the default deep generative approach where one model at a time can be loaded into the GPU and the LM loss for the batch (4) is computed linearly.

Model	Storage Space	GPU Memory	Inference Speed
Unigram SVM	79.6 MB	[CPU]	84x
Lotfi et al. (2020)	15.95 GB	16.6 GB	x
ProDAPT	2.4 GB	4.1 GB	13x

Table 1: Results in terms of the storage space, GPU memory and inference speed.

Model	TOEFL11 (test set)	TOEFL11 10FCV
Unigram SVM	75.8	76.6
Lotfi et al. (2020)	89.0	86.8
ProDAPT	84.2	82.4

Table 2: Results in terms of classification accuracy (%) on the TOEFL11 dataset.

The results presented in Table 1 demonstrate how the ProDAPT architecture optimises the state-of-the-art hardware requirements. The full parallelization support enables a considerable increase of 13x in the inference speed in comparison to the deep generative approach. As the number of supported languages (L1) increases, we predict that this performance gap will widen and become more significant. The unigram SVM baseline proves to be the most lightweight approach, and it provides the best results in terms of the inference speed. However, in order to achieve comparable performance to that of the deep generative approach, other approaches need to make use of ensemble learning and feature engineering. The extensive feature engineering in state-of-the-art systems, which can include hundreds of linguistic features, also comes at the expense of inference and training speed. The results shown in Table 2 indicate that our system outperforms the baseline and achieves an on-par performance with the deep generative approach. We found that the performance decline in our approach was caused by the difficulty of distinguishing between similar language pairs, such as Hindi-Telugu and Japanese-Korean, also noted by Lotfi et al. (2020). We did not observe convergence problems based on the LM loss for any of the L1 models, but we speculate that the restricted number of parameters in the adapters results in a reduced capacity to capture discriminative features that can distinguish between similar languages compared to full model fine-tuning.

6 Conclusion

We proposed an efficient and scalable NLI system based on the state-of-the-art deep generative approach. The method consists of training a transformers adapter for every L1 which can be attached simultaneously to a GPT-2 model for parallel inference. We showed that it is possible to optimize the hardware requirements and the inference speed at the cost of a slight decrease in the model performance.

Acknowledgments

This research received funding through the innovation cheque programme from the Swiss Innovation Agency. It is part of the research collaboration between the University of Zurich and PRODAFT under the project name 59977.1 INNO-ICT "Author profiling to automatize attribution during cybercrime investigations".

References

Johannes Bjerva, Gintarė Grigonytė, Robert Östling, and Barbara Plank. 2017. Neural networks and spelling features for native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 235–239.

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. Toefl11: A corpus of non-native english. *ETS Research Report Series*, 2013(2):i–15.

Lingzhen Chen, Carlo Strapparava, and Vivi Nastase. 2017. Improving native language identification by using spelling errors. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 542–546.

Andrea Cimino and Felice Dell'Orletta. 2017. Stacked sentence-document classifier approach for improving native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 430–437.

Andrea Cimino, Felice Dell'Orletta, Giulia Venturi, and Simonetta Montemagni. 2013. Linguistic profiling based on general–purpose features and native language identification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 207–215.

Vuk Habic, Alexander Semenov, and Eduardo L Pasiliao. 2020. Multitask deep learning for native language identification. *Knowledge-Based Systems*, 209:106440.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Automatically determining an anonymous author's native language. In *International Conference on Intelligence and Security Informatics*, pages 209–217. Springer.

Artur Kulmizev, Bo Blankers, Johannes Bjerva, Malvina Nissim, Gertjan van Noord, Barbara Plank, and Martijn Wieling. 2017. The power of character n-grams in native language identification. In *Proceedings of the 12th workshop on innovative use of NLP for building educational applications*, pages 382–389.

Wen Li and Liang Zou. 2017. Classifier stacking for native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 390–397.

Ehsan Lotfi, Ilia Markov, and Walter Daelemans. 2020. A deep generative approach to native language identification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1778–1783.

Shervin Malmasi, Keelan Evanini, Aoife Cahill, Joel Tetreault, Robert Pugh, Christopher Hamill, Diane Napolitano, and Yao Qian. 2017. A report on the 2017 native language identification shared task. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 62–75.

Shervin Malmasi, Joel Tetreault, and Mark Dras. 2015. Oracle and human baselines for native language identification. In *Proceedings of the tenth workshop on innovative use of NLP for building educational applications*, pages 172–178.

Ilia Markov, Lingzhen Chen, Carlo Strapparava, and Grigori Sidorov. 2017. Cic-fbk approach to native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 374–381.

Terence Odlin. 1989. *Language transfer*, volume 27. Cambridge University Press Cambridge.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. Mad-x: An adapter-based framework for multi-task cross-lingual transfer.

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.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020. Adapterdrop: On the efficiency of adapters in transformers.

Hannah Sterz, Clifton Poth, Andreas Rücklé, and Jonas Pfeiffer. 2021. Adapters for generative and seq2seq models in nlp. *Blog Post*.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the eighth workshop on innovative use of NLP for building educational applications*, pages 48–57.

Ahmet Yavuz Uluslu. 2022. Automatic lexical simplification for Turkish. *arXiv preprint arXiv:2201.05878*.

Arguments to Key Points Mapping with Prompt-based Learning

Ahnaf Mozib Samin **Behrooz Nikandish** **Jingyan Chen**

University of Groningen
Groningen, The Netherlands
{asamin9796, behrooz.nikandish, chenjingyan0722}@gmail.com

Abstract

Handling and digesting a huge amount of information in an efficient manner has been a long-term demand in modern society. Some solutions to map key points (short textual summaries capturing essential information and filtering redundancies) to a large number of arguments/opinions have been provided recently (Bar-Haim et al., 2020). To complement the full picture of the argument-to-keypoint mapping task, we mainly propose two approaches in this paper. The first approach is to incorporate prompt engineering for fine-tuning the pre-trained language models (PLMs). The second approach utilizes prompt-based learning in PLMs to generate intermediary texts, which are then combined with the original argument-keypoint pairs and fed as inputs to a classifier, thereby mapping them. Furthermore, we extend the experiments to cross/in-domain to conduct an in-depth analysis. In our evaluation, we find that i) using prompt engineering in a more direct way (Approach 1) can yield promising results and improve the performance; ii) Approach 2 performs considerably worse than Approach 1 due to the negation issue of the PLM.

1 Introduction

With internet technology getting more accessible to the general public, a flood of information in the digital space can be observed. On online social media, people tend to provide arguments/counter-arguments on various topics, including government policies, movie reviews, and controversial issues such as gun control, abortion, and global climate changes, etc. This kind of information is valuable for government policymakers, business people, and academicians who conduct research on societal changes over time. However, due to the abundance of arguments, it becomes nearly impossible to go through each one manually and make a decision. Moreover, manually reading the arguments does not allow systematic categorization, making it unlikely to quantify them.

To address the issue, (Bar-Haim et al., 2020) first proposed a method to categorize the arguments by mapping them to a set of pre-defined key points set by the domain experts. They fine-tuned a pre-trained language model (PLM) using the ArgKP dataset they built. Fine-tuning PLMs has been proved to achieve superior results over the conventional approach of training a neural network model from scratch. However, there are several limitations to directly fine-tuning PLMs. First, fine-tuning a PLM requires a substantial amount of data and computational resources for each downstream task. Second, the typical way of directly fine-tuning the PLMs does not simulate how the human brain performs NLP tasks. Humans need to be prompted by providing additional task-specific information at first. For example, if we want to know whether a review is positive, negative, or neutral from a human, we would prepare a question like "Do you think the review is positive, negative, or neutral?" to prompt the human to accomplish the task.

Prompt-based learning, built on language models that model the probability of text directly, has been a recent revival in NLP and has shown great potential to address the above limitations. (Brown et al., 2020) indicated that developing a very large PLM with 175 billion tokens and prompting the PLM alleviates the need for additional data for fine-tuning. Thus, it allows us to perform zero-shot and few-shot learning for several NLP tasks. Motivated by this, we exploit prompt-based learning to accomplish the argument-to-keypoint summarization task. More precisely, we would like to shed light on the following research questions:

- Does prompt-based learning allow better utilization of the PLMs for the argument-to-keypoint mapping task? In other words, can it outperform the typical direct fine-tuning PLMs approach?

- What are the challenges that arise with imple-

menting prompt-based learning for this task?

Our contributions are mainly two-fold:

- First, we implement prompt-based learning for the argument-to-keypoint mapping task for the first time, to the best of our knowledge, and compare the results with conventional fine-tuning approaches. To this end, we fine-tune the T5-base PLM with five different prompt templates and report their final F1-scores.
- Second, we propose a novel architecture that takes an argument as input using prompt-based learning and generates an intermediary text after fine-tuning the PLMs. Then, we employ several machine learning classifiers to decide whether the argument, key point, and the intermediary text triple are a match. We demonstrate and analyse the promising results and shortcomings of the proposed architecture.

2 Related Work

Some researchers have done well-executed and rigorous studies and provided thoughtful methods in the field of argument-to-keypoint summarization. (Bar-Haim et al., 2020) established the ArgKP dataset which is the first large-scale dataset for this task and proposed a method to automatically map many arguments to a small number of given key points. They analysed and evaluated some unsupervised methods with TF-IDF and word embeddings and supervised methods like fine-tuning Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018). This study made an excellent basis for next research in this field and is also the foundation of our project. To improve the performance of this task, (Kapadnis et al., 2021) leveraged existing state-of-the-art PLMs along with incorporating additional datasets (IBM Rank 30k and STS) and features like the topic of arguments. But the main shortcoming of these two studies is that the key points are pre-defined by expert annotators, which is an obstacle to making the process fully automatic.

Later, (Bar-Haim et al., 2020) made a more in-depth study to promote the previous line of research, and developed a method for extracting key points automatically from a set of comments, which allows fully automatic key point analysis. And they compared more PLMs including BERT-large-uncased (Devlin et al., 2018), XLNett-large-cased (Yang et al., 2019), RoBERTa-large (Liu et al., 2019), and ALBERT-xxlarge-v1 (Lan et al., 2020), in terms of run time and accuracy, which showed a significant improvement above their previous best results in (Bar-Haim et al., 2020). However, during the step of the automatic key point extraction process, they considered only single sentences and filtered out long sentences as well as those sentences that start with pronouns. Consequently, the model likely misses some potential key points.

Prompt engineering has recently become an emerging field of study in NLP. (Liu et al., 2021) introduced the basics of this new paradigm in detail, and (Brown et al., 2020) confirmed the advantages of adopting prompt-based learning on various NLP tasks such as question answering, translation, and probing tasks for common sense reasoning. And prompt engineering techniques also work well on probing factual knowledge in language models (Jiang et al., 2020). Nonetheless, the suitability of using prompt-based learning for a wide variety of NLP tasks has yet to be proven, and prompt-based learning has not been explored deeply in argument to key point summarization. In this work, we employ this promising paradigm to improve the task of argument-to-keypoint mapping further and provide two approaches to examine the performance of prompt-based learning.

3 Methods

We explore two approaches for this task and compare them with our baselines without any prompt engineering technique. Approach 1 aims to make classification using prompt-based learning. And Approach 2 consists of text generation and text classification with the help of prompt engineering. We use three different Transformer-based (Vaswani et al., 2017) PLMs (BERT (Devlin et al., 2018), BART (Lewis et al., 2020) and T5 (Raffel et al., 2020)) to implement these approaches. The following subsections describe how these PLMs work and why they are appropriate for this task. Moreover, we introduce prompt engineering and the structure of the two approaches in this section.

3.1 Pre-trained Language Models

BERT Unidirectional pre-train architectures limit the choice of architectures during pre-training.

For instance, utilizing left-to-right architecture like in OpenAI GPT (Radford et al., 2018), each token can only attend to previous tokens in the self-attention layer of the Transformer. (Devlin et al., 2018) proposed BERT to alleviate the limitations of unidirectional architectures using *a masked language model*. The architecture of the model is a multi-layer bidirectional Transformer encoder. The model is pre-trained utilizing two unsupervised tasks: Masked Language Models and Next Sentence Prediction (NSP). In many downstream tasks as well as the argument-to-keypoint task, understanding the relationship between two sentences is critical. The BERT model is pre-trained for a binarized NSP task to train the model to understand sentence relationships, which makes the BERT model a good choice for the key point analysis task.

BART BART (Lewis et al., 2020) is a PLM that combines Bidirectional and Auto-Regressive Transformers. The denoising autoencoder is built using a sequence-to-sequence model and it can be applied to various downstream tasks. It uses a standard Transformer-based neural machine translation architecture with a bidirectional encoder and a left-to-right decoder. During the pre-training process, an arbitrary noise function is applied to the input text, and then a sequence-to-sequence model is responsible for reconstructing the original text. Section 3.3.2 describes Approach 2 in which our model generates an intermediary text. BART can be a reasonable choice for this task because it performs effectively in text generation (Yuan et al., 2021) and text summarization (Huang et al., 2020) tasks.

T5 (Raffel et al., 2020) proposed a unified text-to-text Transformer-based model to explore the limitations of transfer learning using an encoder-decoder architecture. It comprises an encoder that maps the input words from the source language to an output representation. The decoder is a conditional language model that attends to the encoder representation and generates target words one by one, based on the source word and previously generated target language words at each time step. The main idea behind this model is to consider all text processing tasks as a text-to-text problem, feeding the model a text as input and generating new text as output. This provides the ability to apply the model, loss function, hyperparameters, and other parameters to various tasks, including machine translation, text summarization and classification, and question answering. We plan to use T5 in both Approach 1 and 2.

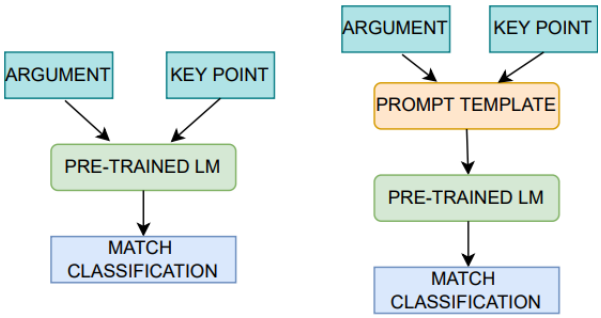


Figure 1: The architecture of baseline (left) and Approach 1 (right)

3.2 Baseline

The architecture of our baseline which is shown in Figure 1 on the left is similar to (Bar-Haim et al., 2020)'s work. We build a classifier to identify whether a pair of *(argument, key point)* is matched or not. To this end, we fine-tune four PLMs, BERT-base, BERT-large, T5-base, and T5-small. We train the models with the train set first and adjust the parameters like epoch with the dev set, and the trained model is evaluated using the unseen data from the test set finally. We do not employ any prompt engineering in the baselines to compare our results with other prompt-based learning approaches in this work.

3.3 Prompt Engineering

To train a model in traditional supervised learning, it is required to have large amounts of supervised data for the task at hand. Prompt-based learning approaches are an attempt to get around this problem. We will first explain the basic form of prompting, and then show how we adopt prompting techniques in the argument-to-keypoint mapping task.
(Liu et al., 2021) described the basic prompting process in three steps: The first step is *prompt addition*, in which a *prompting function* is defined to pre-process the input text. This step consists of two processes:

1. Creating a *template*, which consists of some fixed extra tokens and two slots: *input slot* [X] for input text and *answer slot* [Z] for predicted output that will be used in the *answer mapping* step.

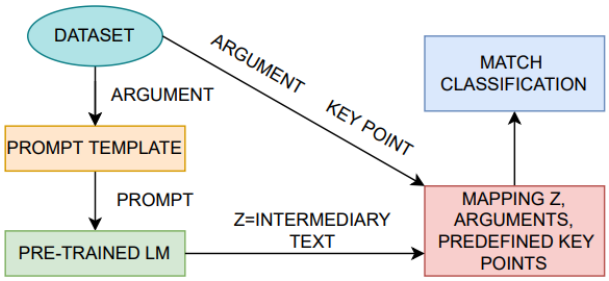


Figure 2: The architecture of Approach 2

2. Filling input slot [X] with the input text.

The output slot [Z] could be either in the middle of the template (*cloze prompt*) or at the end (*prefix prompt*). Depending on the task, the number of input and output slots can vary freely. The second step is *answer search*. In this step, the output slot [Z] in the prompt will be filled by a potential answer, which is the highest scoring answer. In the last step, *answer mapping*, the highest-scoring answer will be mapped to the highest-scoring output. This is the case in text generation tasks, but in some tasks like text classification, each potential answer has a corresponding output to be mapped to.
To answer the research question, we use prompt engineering techniques in two separate approaches for cross/in-domain to investigate if prompt-based learning can outperform our baselines. The architectures of our approaches are discussed in the following subsections.

3.3.1 Approach 1

In Approach 1, we use prompt-based learning in conjunction with fine-tuning a PLM. As the architecture illustrated in Figure 1 on the right, the *(argument, key point)* pairs are transformed to the given prompt template and fed to the T5-base as input. We try various templates shown in table 2 to see how different templates influence our results. In these templates, taking *(argument, key point)* as input texts, [X1] and [X2] represent the *argument* and the *key point* respectively. The answer space of [Z], which is the output text, can be either *matched/not matched* or *Yes/No*, depending on the chosen template. The following example shows the process of creating an input text using a prompt template:

- **Argument [X1]**: *Urbanization destroys the enviroment, and mankind should be finding ways of utilising the space already occupied more efficiently instead*
- **Key point [X2]**: *Urbanization harms the environment*
- **Answer space [Z]**: *matched, not matched*
- **Template**: *The argument: [X1] is [Z] with the key point: [X2]*
- **Input text**: *The argument: urbanization destroys the enviroment, and mankind should be finding ways of utilising the space already occupied more efficiently instead, is* ***matched*** *with the key point: Urbanization harms the environment.*

3.3.2 Approach 2

For Approach 2, we want to explore the effect of adding additional context based on the prior knowledge of the PLMs. Figure 2 illustrates the architecture of Approach 2. First, the input argument is transformed into the given prompt template as [X] and then is fed to the trained PLM (T5-small or BART-large), which is used for text summarization. The output slot [Z] is filled by a generated summary that is called as *intermediary text* in this paper. To note that we use different templates displayed in Table 4 for matching/non-matching pairs to generate the corresponding intermediary texts, and the templates for two types of pairs have totally opposite connotations (e.g., `mean-not mean` and `correct-wrong`). Lastly, different classifiers are built to determine whether the generated intermediary text, argument, and keypoint triple is matching or not. In this step, we fine-tune BERT-base and T5-small PLMs and apply three machine learning algorithms (Naive Bayes (McCallum et al., 1998), Support Vector Machine (SVM) (Cortes and Vapnik, 1995), Decision Tree(Quinlan, 1986)) with TF-IDF features.

4 Experiments

This section will introduce the ArgKP dataset we use and elaborate on how we processed the data and carried out the experiments.

4.1 Data

We use the established ArgKP dataset (Bar-Haim et al., 2020) in this project. The arguments in ArgKP revolve around 28 disputed topics, and they are a subset of the IBM-Rank-30k dataset (Gretz et al., 2020). The key points were authored by an expert on those topics. Crowd annotations

Topic	Argument	Key point	Stance	Label
We should abandon the use of school uniform	we should not abandon the use of school uniforms because it allows children to not be concerned with competitiveness while attempting to learn.	Children can still express themselves using other means	-1	0
We should adopt atheism	we should adopt atheism because religion causes too much tension and disagreements.	Atheism should be adopted since we cannot prove that God exists	1	0
We should end mandatory retirement	mandatory retirement is a good way of refreshing the workforce, motivating those lower in the pecking order and creating employment opportunities.	A mandatory retirement age creates opportunities for other workers	-1	1

Table 1: Examples from the ArgKP dataset. **Stance** means the argument support(1) or oppose(-1) the topic; **Label** represents the key point is matching (1) or non-matching (0) with the argument.

Experiment	Class	Number of samples per set						Total samples	
		Train		Dev		Test			
		Count	(%)	Count	(%)	Count	(%)	Count	(%)
Cross-domain	All	17,019	70.6	2,903	12.1	4,171	17.3	24,093	100.0
	Matching	3,510	14.5	728	3.0	760	3.1	4,998	20.7
	Non-matching	13,509	56.1	2,175	9.1	3,411	14.2	19,095	79.3
In-domain	All	17,021	70.6	2,904	12.1	4,168	17.3	24,093	100.0
	Matching	3563	14.8	593	2.5	842	3.5	4998	20.7
	Non-matching	13458	55.8	2311	9.6	3326	13.8	19,095	79.3

Table 2: Dataset distribution for cross/in-domain experiments

were gathered to see if a keypoint represented or matched an argument, which resulted in *(argument, key point)* pairs. As shown in the Table 1, each pair is assigned a matching or non-matching label and a stance towards the topic.

There are 24,093 labeled argument-keypoint pairs, and 20% of them are matching/positive pairs. Table 2 displays the distribution of each data split set for cross/in-domain experiments. For the cross-domain experiments, we split the whole dataset according to the number of topics, and each topic only occurs once. We assign 19 topics to the train set, and the dev and test sets contain 4 and 5 topics, respectively. The argument-keypoint pair ratio of the three sets is 71:12:17. For the in-domain experiments, we use the same pair ratio of three split sets as the cross-domain experiments, and each split set includes all of those 28 topics.

4.2 Pre-processing

The ArgKP dataset is well-structured and clean enough so that we do not do much pre-processing except for some basic steps. Some arguments and all key points in the dataset do not contain full stops at the end of the sentence, so our first step is to add full stops for each full sentence if they are missing. The second step is tokenization. The PLMs (BERT, BART, T5) we mainly utilize expect a sequence of tokens as an input, so the tokenizers those PLMs were trained on are employed to tokenize the texts. For the machine learning algorithms (SVM, Naive Bayes, Decision Tree), we tokenize the texts and remove stop words using NLTK python package (Bird et al., 2009).

4.3 Experiment Setup

We implement Approach 1 using OpenPrompt (Ding et al., 2021) framework[1], which is an extensible and open-source toolkit for prompt engineering. We replicate their code to train T5-base using the ArgKP dataset for this task. The code associated with this paper is available on a GitHub repository.[2]

Table 3 contains the some of the hyperparameters of each PLM that is fine-tuned in the baselines and Approach 1 and 2.

PLM	Learning Rate	Epoch	Optimizer
BERT-base BERT-large	2e-5	3	Adam
T5-base	1e-3/1e-4	3	Adam
T5-small	3e-4	4	Adam
BART-large	2e-5	5	Adam

Table 3: Hyperparameters used for finetuning different PLMs

4.4 Evaluation

Only about 20% pairs in the dataset are matching/positive pairs, which means the class distribution is quite imbalanced, and standard metrics such as classification accuracy would be misleading in our case. Therefore, we adopt the macro-averaged F1-score, which takes the arithmetic mean of all the per-class F1-scores as the evaluation method.

In addition, we also attempt threshold metrics in order to handle the imbalance problem. Thresholds are learned from the dev set by maximizing the macro-averaged F1-score. Pairs whose matching score exceeds the learned threshold are considered matched. However, we think it is unfair to compare the results of Approach 1/2 and the baselines with different thresholds. Furthermore, most of the learned thresholds are 0.5, which is the same as the default threshold of binary classification. Accounting for these reasons, we ignore threshold metrics finally.

5 Results & Discussion

5.1 Comparison between baseline and the two approaches

Table 4 shows the comparison between our baselines and the two approaches for both in-domain and cross-domain experiments. We have four baselines that do not incorporate prompt engineering. BERT-base outperforms the rest of the four models, getting an F1-score of 88.4% in the in-domain experiment and 72.0% in the cross-domain experiment. For Approach 1, which utilizes prompt engineering and fine-tuning T5-base with the templates, we get higher F1-scores for each of the five prompt templates examined in this study compared to our baselines. Using the five templates, we achieve almost similar F1-scores for the in-domain experiments, while variations in the F1-scores can be observed for the cross-domain evaluation. T1 template obtains the highest F1-scores with 91.4% for the in-domain and 76.1% for the cross-domain experiments. T2 also achieves the second-best F1-score of 91.0% and the equal F1-score to T1. However, T3 and T4 (template with a definition of the key point) can get F1-scores below 74%.

As mentioned in section 3.3.2, our Approach 2 explores T5-small and BART-large by fine-tuning them with two prompt templates (T6 and T7) to get the intermediary texts. Then a classifier decides whether this is a match or non-match based on the argument, intermediary text, and key point as inputs. In the case of the T6 template, with fine-tuning the BART-large for getting the intermediary texts and using BERT-base as a classifier, we achieve the highest F1-scores of 89.2% and 71.2% for in-domain and cross-domain experiments, respectively. But the best-performing system using the T7 template utilizes T5-small to get the intermediary texts and BERT-base and T5-small as classifiers for in-domain and cross-domain experiments, respectively. The final F1-scores from the best-performing model using the T7 template are 90.0% and 69.8% for in-domain and cross-domain experiments, accordingly. This experiment shows that the T6 template is more suitable for BART-large, whereas the T7 template works well with T5-small. The F1-scores using the Naive Bayes, SVM, and Decision Tree as classifiers are poor compared to T5-small and BERT-base.

Comparing the best-performing models of the baselines, Approach 1 and Approach 2, it is evident that Approach 1 outperforms the baseline for both in-domain and cross-domain datasets. Approach 1 also gets substantial improvement in F1-score getting 76.1%, compared to Approach 2, which gets 69.8% for the cross-domain experiment. While the difference in F1-scores between Approach 1 and 2

[1] https://github.com/thunlp/OpenPrompt

[2] https://github.com/samin9796/arg2keypoint

	Prompt Template	PLM for Intermediary Text	Model	F1-score in-domain	F1-score cross-domain
Baseline	-	-	T5-small	0.866	0.700
			T5-base	0.842	0.682
			BERT-base	0.884	0.720
			BERT-large	0.880	0.709
Approach 1	T1: The argument: [X1] and the keypoint [X2] are [Z].	-	T5-base	**0.914**	**0.761**
	T2: The argument: [X1] is [Z] with the keypoint: [X2]			0.910	**0.761**
	T3: Does the argument: [X1] comprise the fact that [X2]? [Z]			0.908	0.732
	T4: A keypoint is a summarization of the corresponding argument. In other words, an argument comprises a keypoint. Does the argument: [X1], comprise the keypoint [X2]? [Z]			0.913	0.737
	T5: Argument: [X1] Keypoint: [X2] $"soft" : "Does"$ $"soft" : "the", "soft_id" : 1$ argument matches $"soft_id" : 1$ keypoint? [Z]			0.911	0.754
Approach 2	T6: [X1] This means [Z1]. $[X1]$ This does not mean [Z1]	T5-small	Naive Bayes	0.493	0.450
			SVM	0.535	0.480
			Decision Tree	0.543	0.498
			T5-small	0.845	0.678
			BERT-base	0.856	0.671
	T6: [X1] This means [Z1]. $[X1]$ This does not mean [Z1]	BART-large	Naive Bayes	0.502	0.527
			SVM	0.674	0.513
			Decision Tree	0.629	0.512
			T5-small	0.830	0.677
			BERT-base	0.892	0.712
	T7: The correct keypoint for the argument: "[X1]" is [Z1] The wrong keypoint for the argument: "[X1]" is [Z1]	T5-small	Naive Bayes	0.485	0.451
			SVM	0.573	0.499
			Decision Tree	0.549	0.501
			T5-small	0.835	0.698
			BERT-base	0.900	0.679
	T7: The correct keypoint for the argument: "[X1]" is [Z1] The wrong keypoint for the argument: "[X1]" is [Z1]	BART-large	Naive Bayes	0.500	0.520
			SVM	0.667	0.529
			Decision Tree	0.614	0.477
			T5-small	0.810	0.678
			BERT-base	0.896	0.664

Table 4: Results of baselines and Approach 1 and 2 for cross/in-domain experiments

for the in-domain experiment is minimal, with Approach 1 getting 91.4% and Approach 2 90.0%. We obtain a higher F1-score using Approach 2 (90%) compared to the baselines (88%) on the in-domain dataset, but on the cross-domain dataset, the F1-score from Approach 2 (69.8%) is lower than the

baseline (72.0%).

5.2 Error Analysis of Approach 2

Table 4 shows that the overall performance of Approach 2 is poor in comparison with Approach 1 for cross/in-domain experiments. We dive into the reason hidden behind this result and make two assumptions. The first assumption is that the language models used for getting the intermediary texts suffer from negation issue. As explained in section 3.3.2, we use slightly different templates for matching/non-matching argument-keypoint pairs to generate their intermediary texts. The templates for non-matching pairs contain a negative connotation like `not` or `wrong`, but the problem is that the PLMs (T5-small and BART-large) cannot capture negation which is demonstrated by the generated intermediary texts being almost the same regardless of whether the template is positive or negative, and thus it results in lower F1-scores from Approach 2. To make it clear, the two following intermediary text examples are extracted from the train set. Given an argument and matching and non-matching key points corresponding to the argument, we can see that their intermediary texts are the same irrespective of using two versions of prompt templates (e.g. positive and negative).

- **Argument**: *by copying something you can not get a pure copy. each copy that is made is worse then the other meaning that no one knows what can happen with cloning.*
- **Keypoints**:
  matching - *Cloning is not understood enough yet*
  non-matching - *Cloning is unethical/anti-religious-*
- **Intermediary texts for both matching/non-matchin pairs**: *Cloning is unnatural*

The second assumption is a decision-making process during the evaluation time. Alluded to previously, the corresponding template is selected based on matching/non-matching labels for each pair in the train set to generate intermediary texts. However, the labels are hidden in the test set, and the specific template can not be chosen. On this account, we always use the non-negative templates (*This means* and *The correct key point*) to get the intermediary text during the inference time.

Even though the overall results of Approach 2 are not as good as we expect, there are still some promising aspects. If the negation issue is solved successfully, Approach 2 could alleviate the need for predefined key points since it can automatically generate texts/key points.

6 Conclusions and Future Work

In this work, we first build the baseline models for the argument to keypoint mapping task by finetuning PLMs without implementing prompt engineering. Then, we take advantage of prompt-based learning and utilize it while finetuning PLMs with two different approaches. From the comparison between the baselines and the two specific approaches, prompt engineering substantially improves the performance of the task. However, it still includes some challenges and limitations that need to be investigated more in the case of Approach 2. To be more specific, in Approach 1, we attempt five different prompt templates with T5-base, and all of the results are better than the baselines for both cross/in-domain experiments. While in Approach 2, the intermediary texts generated from T5-small and BART-large using two prompt templates reduce the overall performance compared to baselines.

The mapping of arguments to key points can be viewed as an intermediate step toward fully automatic argument summarization. Therefore, in future work, we plan to tackle the negation problem of PLMs in Approach 2, which would be promising for generating key points automatically. Furthermore, experimenting with other sequence-to-sequence models using prompt-based learning is another interesting future direction.

Acknowledgement

The authors would like to thank Dr. Khalid Al-Khatib of the University of Groningen, The Netherlands for his support and assistance.

References

Bar-Haim, R., L. Eden, R. Friedman, Y. Kantor, D. Lahav, and N. Slonim (2020). From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, pp. 4029–4039. Association for Computational Linguistics.

Bar-Haim, R., Y. Kantor, L. Eden, R. Friedman, D. Lahav, and N. Slonim (2020). Quantitative argument summarization and beyond: Cross-domain key point

analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, pp. 39–49. Association for Computational Linguistics.

Bird, S., E. Klein, and E. Loper (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems 33*, 1877–1901.

Cortes, C. and V. Vapnik (1995). Support-vector networks. *Machine learning 20*(3), 273–297.

Devlin, J., M. Chang, K. Lee, and K. Toutanova (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR abs/1810.04805*.

Ding, N., S. Hu, W. Zhao, Y. Chen, Z. Liu, H.-T. Zheng, and M. Sun (2021). Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998*.

Gretz, S., R. Friedman, E. Cohen-Karlik, A. Toledo, D. Lahav, R. Aharonov, and N. Slonim (2020). A large-scale dataset for argument quality ranking: Construction and analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 34, pp. 7805–7813.

Huang, D., L. Cui, S. Yang, G. Bao, K. Wang, J. Xie, and Y. Zhang (2020, November). What have we achieved on text summarization? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, pp. 446–469. Association for Computational Linguistics.

Jiang, Z., A. Anastasopoulos, J. Araki, H. Ding, and G. Neubig (2020, November). X-FACTR: Multilingual factual knowledge retrieval from pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, pp. 5943–5959. Association for Computational Linguistics.

Kapadnis, M., S. Patnaik, S. Panigrahi, V. Madhavan, and A. Nandy (2021, November). Team enigma at ArgMining-EMNLP 2021: Leveraging pre-trained language models for key point matching. In *Proceedings of the 8th Workshop on Argument Mining*, Punta Cana, Dominican Republic, pp. 200–205. Association for Computational Linguistics.

Lan, Z., M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut (2020). Albert: A lite bert for self-supervised learning of language representations. In *ICLR*. OpenReview.net.

Lewis, M., Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer (2020, July). 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*, Online, pp. 7871–7880. Association for Computational Linguistics.

Liu, P., W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig (2021). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ArXiv abs/2107.13586*.

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov (2019). Roberta: A robustly optimized bert pretraining approach. *ArXiv abs/1907.11692*.

McCallum, A., K. Nigam, et al. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, Volume 752, pp. 41–48. Citeseer.

Quinlan, J. R. (1986, March). Induction of decision trees. *Mach. Learn. 1*(1), 81–106.

Radford, A., K. Narasimhan, T. Salimans, and I. Sutskever (2018). Improving language understanding by generative pre-training.

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. 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.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Volume 30. Curran Associates, Inc.

Yang, Z., Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Volume 32. Curran Associates, Inc.

Yuan, W., G. Neubig, and P. Liu (2021). Bartscore: Evaluating generated text as text generation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems*, Volume 34, pp. 27263–27277. Curran Associates, Inc.

Pre-training Language Models for Surface Realization

Farhood Farahnak
Concordia University
Montreal, Canada
farhood.farahnak@gmail.com

Leila Kosseim
Concordia University
Montreal, Canada
leila.kosseim@concordia.ca

Abstract

Surface Realization in Natural Language Generation (NLG) is the task of deriving the surface form of a sentence (the actual words) from an underlying representation. Following recent advances in deep learning, several models have been proposed for different NLG sub-tasks including surface realization. Most of these models require a large amount of training data, however, acquiring accurately labeled data is laborious and expensive. In this work, we study how synthetically generated labeled data can be leveraged to improve the performance of a surface realization model. By pre-training a language model on automatically labeled data and then fine-tuning it on manually labeled data, our approach improved the state-of-the-art performance on the standard English datasets from the deep track of the Multilingual Surface Realization (MSR) workshop (Belz et al., 2020) by more than 10% BLEU score.[1]

1 Introduction

The goal of Natural Language Generation (NLG) is to generate text in human languages (e.g. English) for a wide range of applications such as report generation, text summarization, and conversation modeling. NLG involves both *content planning* (selecting the content to communicate) and *surface realization*. Surface realization (SR), the last step of the NLG pipeline, aims to derive the surface form of a sentence (the actual words) from an underlying representation by choosing the proper word forms (inflection, punctuation, and formatting) and determining their correct order (syntactic realization) (Hovy et al., 1996; Reiter and Dale, 2000).

Recent advances in Natural Language Processing (NLP) and Deep Neural Networks (DNN) have led to drastic improvements in many NLP systems, some of which have even achieved human-level performance (Läubli et al., 2018). Similarly to many NLP models, surface realization models have also benefited from these advancements. DNN models usually require a large amount of labeled data for training; however, creating accurate and reliable training data is an expensive and time consuming task. In this work, we show how we can improve the performance of surface realization by pre-training a language model on a large synthetically generated dataset and then fine-tuning it on a smaller manually labeled dataset.

To measure the effectiveness of our approach, we followed the protocol of the Multilingual Surface Realization (MSR) Workshops (Mille et al., 2018, 2019; Belz et al., 2020), and generated the surface form of sentences from their dependency parse trees. To create the synthetic data, we used the automatic dependency parser Stanza (Qi et al., 2020) to parse the unlabeled WikiText corpus (Merity et al., 2017). Using different sizes of manually labeled and synthetic data, we investigated the effects of the proposed pre-training phase. Although the synthetic data may contain noisy annotations compared to manually labeled data and may come from a different distribution (e.g. different textual genre or discourse domain), results show that its sheer size allows the model to learn the general gist of the task in the pre-training phase and leads to an increase in performance in SR achieving state-of-the-art performance on the deep track with the English datasets of the MSR workshops.

2 Background

2.1 Multilingual Surface Realization (MSR)

The Multilingual Surface Realization (MSR) workshops have organized shared tasks aimed at bring-

[1]The code is available at https://github.com/CLaC-Lab/SR_LM

ing together researchers interested in surface oriented Natural Language Generation problems and share resources to that end (Mille et al., 2018, 2019; Belz et al., 2020). The shared task aimed to generate the surface form of sentences given their Universal Dependency (UD) structures. Two tracks were proposed: the shallow and the deep tracks. For the shallow track, word order information and the inflected form of words were removed from the UD structure and the task aimed to determine the correct order of words and inflect them. In the deep track, in addition to word ordering and inflection, functional words (in particular, auxiliaries, functional prepositions and conjunctions) and surface-oriented morphological information were removed from the UD structure and had to be recovered by the models.

2.2 Previous Work

Participants in the Multilingual Surface Realization (MSR) workshops proposed different models to address the surface realization task. Many of these models use dedicated sub-modules for each subtask. For example the ADAPT center (Elder, 2020) proposed a biLSTM sequence-to-sequence model with a copy mechanism to generate the surface form of sentences. They augmented the training set with 4.5M sentences from two sources, WikiText (Merity et al., 2017) and CNN stories (Hermann et al., 2015), and chose sentences that had at least 80% word overlap with the labeled dataset to ensure that they have a similar distribution. The BME-TUW system (Recski et al., 2020) used an Interpreted Regular Tree Grammar to retrieve the correct order of tokens then used a biLSTM sequence-to-sequence model to inflect the words. The IMS system (Yu et al., 2020) tackled the surface realization problem as a Traveling Salesperson Problem, and used a biaffine attention model to calculate the bigram scores for the output sequence. Finally, they used a biLSTM for the inflection module. Similarly to the ADAPT center, IMS also used WikiText and CNN stories to augment their training data with $200K$ synthetic samples, however, by considering the branching factors of the tree, they tried to keep the distribution of the augmented data close to the labeled datasets. The data augmentation that ADAPT and IMS used differ from our proposed solution as they both tried to keep the distribution of the augmented data as similar as possible to the manually labeled data by applying filtering rules. In contrast, our approach does not enforce the distributions to be similar, and lets the domain adaptation to be performed automatically during the fine-tuning phase.

Because of their simplicity and effectiveness, several approaches have used language models for surface realization. The NILC system (Cabezudo and Pardo, 2020) proposed to use GPT-2 (Radford et al., 2019) and linearization using the parentheses approach. We argue that when the number of nodes grows, the model has difficulties in capturing the relations between them. The Concordia system (Farahnak et al., 2020) used BART (Lewis et al., 2020) for surface realization, however, the relation between nodes was represented with the actual words. This approach may cause problems when a word appears more than once in a sentence as the model cannot capture the exact structure of the tree. Our approach is also based on language models, however, it differs from theirs as indices are used to encode the edges in the UD structure instead of the actual tokens (see Section 4).

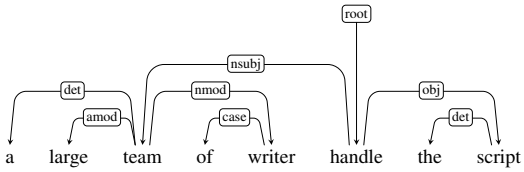


Figure 1: Example of UD dependency parse tree for the sentence *A large team of writers handled the script*.

3 Data

In this section, we first present the MSR manually labeled datasets we used for our experiments and then discuss how we created the synthetic dataset.

3.1 Manually Labeled Datasets

For our experiments, we used the English datasets provided by the MSR workshop (Belz et al., 2020). These datasets are modified versions of the Universal Dependency (UD) datasets (de Marneffe et al., 2014) where the order of the tokens is shuffled and the inflected form of the tokens are removed. Table 1 presents statistics of these datasets. As Table 1 shows, the largest dataset (EWT) contains only $\approx 12K$ training samples which makes it hard to train an DNN model based solely on these samples.

Dataset	train	dev	test
EWT	12,543	2,002	2,077
GUM	2,914	707	778
LinES	2,738	912	914
ParTUT	1,781	156	153

Table 1: Number of samples in the English MSR datasets.

3.2 Synthetically Generated Dataset

In order to generate synthetic data, we used the WikiText dataset (Merity et al., 2017), extracted from Wikipedia articles. The WikiText dataset comes from a different domain compared to the MSR datasets[2] which makes it a suitable candidate to study the domain adaptation between the two text genres. We extracted the first $500K$ sentences after filtering non-English sentences and sentences longer than 150 characters[3] to create our synthetic dataset. We used Stanza (Qi et al., 2020) to parse the sentences and create their UD structure. Using the script provided by the MSR workshop (Belz et al., 2020), we generated the synthetic dataset in the same format as provided by the workshop. Figure 1 shows a visual representation of the dependency tree structure of a sample from the dataset.

4 Model

Following the success of pre-trained language models (LMs) for data-to-text generation tasks (Kale and Rastogi, 2020; Harkous et al., 2020; Farahnak et al., 2020), we used an encoder-decoder LM for surface realization. The input and output of an encoder-decoder LM is in linear form (text-to-text); however, surface realization is a data-to-text task. In order to use LM, the input UD structure had to be linearized. Among the features available in the UD structure, we considered `lemma` (the lemmatized form of tokens), `FEATS` (morphological information), `HEAD` (the parent in the tree structure), and `deprel` (dependency relation to the head) and represented each node in the linear format:

`index : lemma FEATS : head_index <deprel>`

then concatenated all nodes together. Figure 2 shows the linearized representation of the example from Figure 1 used for the shallow track. In this example, the parent of the word `script` is node 5 which is the index for word `handle`. To train the LM, we used the surface form of the sentence as the target. The model learns to generate the surface form given the linearized UD structure, hence, it learns to perform both syntactic and morphological realization simultaneously.

> 4 : script Sing : 5 <obj> # 3 : writer Plur : 7 <nmod> # 9 : . : 5 <punct> # 6 : large Pos : 7 <amod> # 7 : team Sing : 5 <nsubj> # 5 : handle Ind Plur 3 Past Fin : ROOT <root> # 1 : the Def Art : 4 <det> # 8 : of : 3 <case> # 2 : a Ind Art : 7 <det>

Figure 2: Linearized representation of the UD structure of Figure 1.

5 Experiments and Results

5.1 Experimental Setup

In order to understand the effect of synthetic data on the performance of the ordering model, we conducted several experiments using different sizes of synthetic data to pre-train the model, then fine-tuning it on the manually labeled datasets and measuring the performance on the MSR test sets (see Table 1). For all experiments, we used the pre-trained BART (Lewis et al., 2020) large model. We used the AdamW (Loshchilov and Hutter, 2019) optimization algorithm with a learning rate of $1e$-5 and batch size of 4 to train our models. We pre-trained the models for 5 epochs on the synthetic data and fine-tuned them for 5 more epochs on the manually labeled data. For comparative purposes, we also trained the models without the pre-training phase, and trained them for 15 epochs on the manually labeled data. We choose the model with the highest performance on the development sets.

5.2 Results

Table 2 compares the performance of training the encoder-decoder language model using different sizes of synthetic data for pre-training. Our experiments suggest that the pre-training phase can improve the performance of the model by 3.90% and 5.21% in BLEU score for the shallow and deep tracks respectively on the EWT dataset. However, the improvement on the other three datasets are more significant, ranging from 12.76% to 25.65%, as these datasets have much fewer training samples compared to EWT. The improvement of pre-training on synthetic data is higher for the deep

[2]The EWT dataset contains sentences from five genres of web media: weblogs, newsgroups, emails, reviews, and Yahoo! answers.

[3]This value was chosen because 90% of the samples in EWT are shorter than 150 characters.

# synthetic samples for pre-training	Shallow Track								Deep Track							
	EWT	Δ	GUM	Δ	LinES	Δ	ParTUT	Δ	EWT	Δ	GUM	Δ	LinES	Δ	ParTUT	Δ
0	80.79	_	71.63	_	69.62	_	67.84	_	64.31	_	48.74	_	40.61	_	49.23	_
$100K$	84.38	3.59	86.27	14.64	82.38	12.76	86.98	19.14	68.10	3.79	68.61	19.87	64.28	23.67	69.50	20.27
$200K$	84.62	3.83	86.84	15.21	83.00	13.38	86.69	18.85	69.02	4.71	69.21	20.47	65.29	24.65	69.25	20.02
$500K$	**84.69**	3.90	**86.76**	15.13	**83.18**	13.56	87.66	19.82	**69.52**	5.21	**70.19**	21.45	**66.26**	25.65	**71.38**	22.15

Table 2: BLEU score of models pre-trained with different sizes of synthetic data. Δ reports the difference of the pre-trained models to training without the pre-training phase (i.e. 0 synthetic data).

		EWT			GUM			LinES			ParTUT		
		BLEU	NIST	DIST	BLEU	NIST	DIST	BLEU	NIST	DIST	BLEU	NIST	DIST
Shallow Track	BME (Recski et al., 2020)	57.25	12.52	65.23	60.77	12.10	62.86	55.98	11.78	61.44	61.37	10.22	58.39
	Concordia (Farahnak et al., 2020)	70.71	12.70	77.94	66.98	11.62	69.87	62.70	11.30	68.62	67.05	9.83	71.59
	IMS (Yu et al., 2020)	85.67	13.74	87.74	**89.70**	**12.98**	**91.97**	**85.30**	**12.97**	**86.48**	**89.37**	**11.05**	**88.73**
	ADAPT (Elder, 2020)	**87.50**	**13.81**	**90.35**	_	_	_	_	_	_	_	_	_
	Our Approach	84.69	13.58	88.82	86.76	12.65	89.12	83.18	12.59	85.72	87.66	10.91	86.80
Deep Track	NILC (Cabezudo and Pardo, 2020)	45.19	9.96	64.83	53.92	9.00	60.42	41.04	9.09	61.18	43.41	8.24	59.74
	Concordia (Farahnak et al., 2020)	58.44	11.61	73.66	53.92	10.51	67.02	47.96	9.93	64.33	50.54	8.57	62.39
	IMS (Yu et al., 2020)	58.66	11.61	79.23	53.92	11.25	76.47	50.45	10.89	73.1	50.11	9.26	72.98
	Our Approach	**69.52**	**12.54**	**82.43**	**70.19**	**11.64**	**80.93**	**66.26**	**11.37**	**78.81**	**71.38**	**9.99**	**77.88**

Table 3: Comparison of our approach (models pre-trained on $500K$ synthetic sentences and fine-tuned on each dataset) with previous models proposed for the deep track of MSR 2020.

track compared to the shallow track as the task is more complex in the sense that the model not only needs to learn the inflection and ordering of words, it also needs to guess the removed functional words.

In comparison with previous participating models of MSR 2020 (Belz et al., 2020) (see Table 3), our approach is not able to outperform the previous work on the shallow track. However, it improves the state-of-the-art performance by a large margin (more than 10% in BLEU score) on in the deep track on all datasets which shows the superiority of our proposed approach.

5.3 Analysis

We analysed the results of the models to better understand the benefits and drawbacks of our approach.

Pre-training seems to facilitate domain adaption, as a single epoch of fine-tuning is enough for the model to adapt to the domain of the manually labeled dataset (see Appendix A.1).

Pre-training can significantly reduce the need for manual data. We fine-tuned the pre-trained models using subsets of the manually labeled data. Results shows that with pre-training, using only 10% of the data achieves better performance than training on all manually labeled data without the pre-training phase (see Appendix A.2).

Finally, through a manual inspection of the generated sentences (see Appendix A.3), we determined that most errors should actually be considered correct alternatives to the ground truth. Better automatic measures should be developed to measure the performance of surface realization to account for linguistic variations.

6 Conclusion and Future Work

In this paper, we showed that pre-training on synthetic data is beneficial for surface realization even when the data comes from a different distribution than the training data. We also showed that the pre-training phase not only improves the performance of the model, but also helps the model to converge faster on the training data. The proposed pre-training phase for LM improved the state-of-the-art performance on the standard English datasets from the deep track of the MSR workshop (Belz et al., 2020) by more than 10% BLEU score.

As of future work, we plan to conduct similar experiments on previously proposed models such as ADAPT (Elder, 2020) and IMS (Yu et al., 2020). We also plan to run cross-language experiments to see whether the knowledge learned from one language can be transferred to another language.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments on an earlier version of this paper. This work was financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Anya Belz, Bernd Bohnet, Thiago Castro Ferreira, Yvette Graham, Simon Mille, and Leo Wanner. 2020. Proceedings of the Third Workshop on Multilingual Surface Realisation. Barcelona, Spain (Online). Association for Computational Linguistics.

Marco Antonio Sobrevilla Cabezudo and Thiago Pardo. 2020. NILC at SR'20: Exploring pre-trained models in surface realisation. In *Proceedings of the Third Workshop on Multilingual Surface Realisation*, pages 50–56, Barcelona, Spain (Online). Association for Computational Linguistics.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 4585–4592, Reykjavik, Iceland. European Languages Resources Association (ELRA).

Henry Elder. 2020. ADAPT at SR'20: How preprocessing and data augmentation help to improve surface realization. In *Proceedings of the Third Workshop on Multilingual Surface Realisation*, pages 30–34, Barcelona, Spain (Online). Association for Computational Linguistics.

Farhood Farahnak, Laya Rafiee, Leila Kosseim, and Thomas Fevens. 2020. Surface realization using pretrained language models. In *Proceedings of the Third Workshop on Multilingual Surface Realisation*, pages 57–63, Barcelona, Spain (Online). Association for Computational Linguistics.

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, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1693–1701, Cambridge, MA, USA. MIT Press.

Eduard Hovy, Gertjan van Noord, Guenter Neumann, and John Bateman. 1996. Language generation. *Survey of the State of the Art in Human Language Technology*, pages 131–146.

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.

Samuel Läubli, Rico Sennrich, and Martin Volk. 2018. Has machine translation achieved human parity? a case for document-level evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4791–4796, Brussels, Belgium. 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.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*, Ernest N. Morial Convention Center, New Orleans.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, (ICLR 2017), Conference Track Proceedings*, Toulon, France. OpenReview.net.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, and Leo Wanner. 2019. Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019). Hong Kong, China. Association for Computational Linguistics.

Simon Mille, Anja Belz, Bernd Bohnet, Emily Pitler, and Leo Wanner. 2018. Proceedings of the First Workshop on Multilingual Surface Realisation. Melbourne, Australia. 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*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Gábor Recski, Ádám Kovács, Kinga Gémes, Judit Ács, and Andras Kornai. 2020. BME-TUW at SR'20: Lexical grammar induction for surface realization. In *Proceedings of the Third Workshop on Multilingual Surface Realisation*, pages 21–29, Barcelona, Spain (Online). Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Natural Language Processing. Cambridge University Press, New York, NY, USA.

Xiang Yu, Simon Tannert, Ngoc Thang Vu, and Jonas Kuhn. 2020. IMSurReal too: IMS in the surface realization shared task 2020. In *Proceedings of the Third Workshop on Multilingual Surface Realisation*, pages 35–41, Barcelona, Spain (Online). Association for Computational Linguistics.

A Detailed Analysis

A.1 Domain Adaptation

Figure 3 compares the BLEU scores of training models for the deep track on the EWT dataset with and without pre-training on $500K$ synthetic samples with different training epochs. As the figure shows, for the pre-trained model, the domain adaptation phase is almost completed after the first epoch while the non-pre-trained model continues to improve even after 10 epochs.

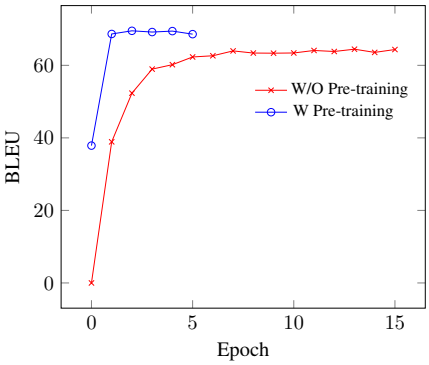


Figure 3: The BLEU score on the EWT test set for all epochs when training the models for the deep track, with and without pre-training on $500K$ synthetic samples.

A.2 Size of Training Data

Table 2 shows that pre-training on synthetic data before fine-tuning on manually labeled data can improve the overall performance of the model. In order to better understand the importance of the size of manually labeled training data, we limited its size and fine-tuned the model on different sizes of manually labeled training data. Table 4 shows the performances of the model after fine-tuning on different subsets of the EWT dataset. As Table 4 shows, fine-tuning solely on $1K$ samples can achieve better performance compared to no pre-training and using the full EWT dataset (last row of Table 4). However, by increasing the number of training samples (from $1K$ to $12.5K$), we can achieve a higher performance when pre-training. This indicates that even though the pre-training phase is helpful for the task, it is not sufficient to replace the manually labeled training data altogether.

# synthetic samples	# manually labeled samples	BLEU	NIST	DIST
$500K$	0	37.87	8.09	61.04
$500K$	1K	64.54	11.88	78.50
$500K$	2K	65.70	12.00	78.93
$500K$	5K	67.35	12.33	80.76
$500K$	10K	68.79	12.43	81.80
$500K$	12.5K	69.52	12.54	82.43
0	12.5K	64.31	11.64	77.80

Table 4: Comparison of the performance of the encoder-decoder model using different sizes of training data for the fine-tuning on a model pre-trained with $500K$ synthetic samples.

A.3 Error Analysis

We manually inspected the errors generated by our models. While a few generated sentences did contained true errors, most can be regarded as correct alternatives to the ground truth. Table 5 shows a few examples. One common correct alternative was related to the generation of contractions as in Ex. 1. This type of error occurs because the MSR input structure of the token to generate (*it*) does not contain any feature that give the model a clue as to whether the token should be contracted or not. In Ex. 2, the model failed to generate the expected punctuation in the deep track, yet the generated sentence is a correct alternative to the ground truth. In Ex. 3, the word order in the generated outputs is not identical to the ground truth; however, they are grammatically correct and convey the same meaning. In Ex. 4, the output of the shallow model is indeed a true error as it is not grammatically correct; however, the deep model generated a grammatically correct output but again it is not identical to the expected output. Finally, in Ex. 5 and 6, show examples of correct alternative to number formatting compared to the ground truth.

Ex. 1	Ground Truth	i 'll post highlights ...
	Output of Shallow	i will post highlights ...
	Output of Deep	i will post highlights ...
Ex. 2	Ground Truth	two weeks later , and the violence continues .
	Output of Shallow	two weeks later , and the violence continues .
	Output of Deep	two weeks later and the violence continues .
Ex. 3	Ground Truth	they own blogger , of course .
	Output of Shallow	of course , they own blogger .
	Output of Deep	of course they own blogger .
Ex. 4	Ground Truth	we have this report ?
	Output of Shallow	have we this report ?
	Output of Deep	do we have this report ?
Ex. 5	Ground Truth	compensation : $ 60000 - 70000
	Output of Shallow	compensation : $ 60,000 - 70,000
	Output of Deep	compensation : $ 60000 - 70000
Ex. 6	Ground Truth	... said that there was a 10 to 50 % chance ...
	Output of Shallow	... said that there was a 10 to 50 % chance ...
	Output of Deep	... said there was a 10 - 50 % chance ...

Table 5: Sample errors generated by the shallow and deep models pre-trained on $500K$ synthetic data and fine-tuned on the EWT dataset.

The 5th International Conference on Natural Language and Speech Processing (ICNLSP 2022)

December 16-17, 2022

Technical program

All times are according to time (GMT)

Friday, Dec. 16, 2022 08:00 – 17:50 (GMT)	
08:00-08:30	*Opening session* **Dr. Mourad Abbas and Dr. Abed Alhakim Freihat**
08:30 – 09:10	*Keynote 1* **:** Plug-and-Play Abilities for Neural Machine Translation **Prof. Jan Niehues**, Karlsruhe Institute of Technology, Germany
09:30 – 11:50	*Oral Session 1 : Dialog systems, natural language understanding, and NLP tools* **Chair: Dr. Tamara Matthews**
09 :30 – 09:45	Stefan Constantin and Alex Waibel. ***Karlsruhe Institute of Technology, Germany.*** **Error-correction and extraction in request dialogs.**
09:50 – 10:05	Radostin Cholakov and Todor Kolev. ***Obecto Ltd., Bukgaria.*** **Efficient Task-Oriented Dialogue Systems with Response Selection as an Auxiliary Task.**
10:10 – 10:25	Hongru Wang, Mingyu Cui, Zimo Zhou and Kam-Fai Wong. ***The Chinese University of Hong Kong, China.*** **TopicRefine: Joint Topic Prediction and Dialogue Response Generation for Multi-turn End-to-End Dialogue System.**
10:30 – 10:45	Zezhong Wang, Hongru Wang, Wai Chung Kwan and Kam-Fai Wong. ***The Chinese University of Hong Kong, China.*** **Prior Omission of Dissimilar Source Domain(s) for Cost-Effective Few-Shot Learning.**
10:50 – 11:05	Zhengxiang Wang. ***Stony Brook University, USA.*** **Linguistic Knowledge in Data Augmentation for Natural Language Processing: An Example on Chinese Question Matching.**
11:10 – 11 :25	Andrea Stefanoni, Šarūnas Girdzijauskas, Christina Jenkins, Zekarias T. Kefato, Licia Sbattella, Vincenzo Scotti and Emil Wåreus. ***Politecnico di Milano, Italy.*** **Detecting Security Patches in Java Projects Using NLP Technology.**

11:30 – 11 :45	Pius von Däniken, Jan Deriu, Eneko Agirre, Ursin Brunner, Mark Cieliebak and Kurt Stockinger, ***Zurich University of Applied Sciences.*** **Improving NL-to-Query Systems through Re-ranking of Semantic Hypothesis.**
11:50 - 13:00	**Break**
13:00 – 13:40	*Keynote 2 :* Hybrid natural language processing in the deep learning era **Prof. Eric Laporte,** Gustave Eiffel University, France
14:00– 15:40	*Oral Session 2: Classification* **Chair: Dr. Muhammad Al-Qurishi**
14:00 – 14:15	Tamara Matthews and David Lillis. ***University College Dublin, Ireland.*** **Experimenting with ensembles of pre-trained language models for classification of custom legal datasets.**
14:20 – 14:35	Yousef Younes and Brigitte Mathiak. ***GESIS -- Leibniz-institute for the Social Sciences, Germany.*** **Handling Class Imbalance when Detecting Dataset Mentions with Pre-trained Language Models.**
14:40 – 14:55	Hlynur Hlynsson, Steindór Ellertsson, Jon Dadason, Emil Sigurdsson and Hrafn Loftsson. ***Reykjavik University, Iceland.*** **Semi-supervised Automated ICD Coding.**
15:00 – 15:15	Emmanuelle Kelodjoue, Jérôme Goulian and Didier Schwab. ***LIG (Laboratoire d'Informatique de Grenoble), France.*** **Performance of two French BERT models for French language on verbatim transcripts and online posts.**
15:20 – 15:35	Muhammad Al-Qurishi, ***Elm, KSA.*** Recent Advances in Long Documents Classification Using Deep-Learning
15:40 - 15:50	**Break**
15:50 – 17:50	*Oral Session 3 : Semantics and Textual Similarity* **Chair: Dr. Muhammad Al-Qurishi**
15:50 – 16:05	Jarkko Lagus and Arto Klami. ***University of Helsink, Finland.*** **Optimizing singular value based similarity measures for document similarity comparisons.**
16:10 – 16:25	Besher Alkurdi, Hasan Yunus Sarioglu and Mehmet Fatih Amasyali. ***Yildiz Technical University, Turkey.*** **Semantic Similarity Based Filtering for Turkish Paraphrase Dataset Creation.**
16:30 – 16:45	Jarkko Lagus, Niki Loppi and Arto Klami. ***University of Helsink, Finland.*** **Second-order Document Similarity Metrics for Transformers.**
16:50 – 17:05	Phillip Schneider, Markus Voggenreiter, Abdullah Gulraiz and Florian Matthes. ***Technical University of Munich, Germany.*** **Semantic Similarity-Based Clustering of Findings From Security Testing Tools.**
17:10 – 17:25	Kyra Wilson and Alec Marantz. ***New York University Abu Dhabi, UAE.*** **Contextual Embeddings Can Distinguish Homonymy from Polysemy in a Human-Like Way.**
17:30 – 17:45	Sagar Indurkhya, ***Massachusetts Institute of Technology, USA.*** **Modeling the Ordering of English Adjectives using Collaborative Filtering.**

	Saturday, Dec. 17, 2022 08:30 – 17:45 (GMT)
08:30 – 09:10	*Keynote 3* **:** Multilingual and Code-Switching Speech Recognition **Dr. Ahmed Ali**, Qatar Computing Research Institute, Qatar
09:30– 11:40	*Oral Session 4: Speech recognition, speaker recognition, and speech synthesis* **Chair: Dr. Ahmed Ali**
09 :30 – 09:45	Laurence Dyer, Anthony Hughes, Dhwani Shah and Burcu Can. ***University of Wolverhampton, United Kingdom.*** **Comparison of Token- and Character-Level Approaches to Restoration of Spaces, Punctuation, and Capitalization in Various Languages.**
10:05 – 10:20	Punnoose Kuriakose. ***Flare Speech Systems, India.*** **New Features for Discriminative Keyword Spotting.**
10:25 – 10:40	Kevin Glocker and Munir Georges. ***Technische Hochschule Ingolstadt, Germany.*** **Hierarchical Multi-Task Transformers for Crosslingual Low Resource Phoneme Recognition.**
10:45 – 11:00	Patrick Donnelly. ***Oregon State University, USA.*** **Concatenative Phonetic Synthesis for the Proto-Indo-European Language.**
11:05 – 11:20	Yu Zhou, B. Chandra Mouli and Vijay Gurbani. ***Vail Systems Inc., USA.*** **A low latency technique for speaker detection from a large negative list.**
11:25 – 11:40	Sreepratha Ram and Hanan Aldarmaki. ***United Arab Emirates University, UAE.*** **Supervised Acoustic Embeddings And their Transferability Across Languages.**
11:45 - 13:00	**Break**
13:00– 15:00	*Oral Session 5: Sentiment Analysis and Dialects* **Chair: Dr. Abed Alhakim Freihat**
13:00 – 13:15	Hend Alkhalifa, Fetoun Alzahrani, H. Qawara, R. Alrowais, S. Alowa and L. Aldhubayi. ***king Saud University, KSA.*** **A Dataset for Detecting Humor in Arabic Text.**
13:20 – 13:35	Emna Fsih, Rahma Boujelbane and Lamia Hadrich Belguith. ***ANLP Research Group / Sfax, Tunisia.*** **A deep sentiment analysis of Tunisian dialect comments on multi-domain posts in different social media platforms.**
13:40 – 13:55	Abir Messaoudi, Hatem Haddad, Moez Benhaj Hmida and Mohamed Graiet. ***iCompass, Tunisia.*** **Toward a Tunisian Speech Emotion Recognition System.**
14:00 – 14:15	Patrick Donnelly and Aidan Beery. ***Oregon State University, USA.*** **Evaluating Large-Language Models for Dimensional Music Emotion Prediction from Social Media Discourse.**
14:20 – 14:35	Dhuha Alqahtani, Lama Alzahrani, Maram Bahareth, Nora Alshameri, Hend Al-Khalifa and Luluh Aldhubayi. ***king Saud University, KSA.*** **Customer Sentiments Toward Saudi Banks During the Covid-19 Pandemic.**

14:40 – 14:55	Khaled Lounnas, Mohamed Lichouri and Mourad Abbas. ***University of Sciences and Technologies, Algeria.*** **Towards an Automatic Dialect Identification System for Algerian Dialects Using YouTube Videos.**
15:00– 16:20	*Oral Session 6 : Data, Information extraction, and summarization* **Chair: Prof. Hend Alkhalifa**
15:00 – 15:15	Kun Sun and Rong Wang. ***University of Tübingen, Germany.*** **Constructing the Corpus of Chinese Textual 'Run-on' Sentences (CCTRS): Discourse Corpus Benchmark with Multi-layer Annotations.**
15:20 – 15:35	Mingyang Song, Yi Feng and Liping Jing. ***Beijing Jiaotong University, China.*** **Utilizing BERT Intermediate Layers for Unsupervised Keyphrase Extraction.**
15:40 – 15:55	Michel Schwab, Robert Jäschke and Frank Fischer. ***Humboldt Universität zu Berlin, Germany.*** **"Der Frank Sinatra der Wettervorhersage": Cross-Lingual Vossian Antonomasia Extraction.**
16:00 – 16:15	Martin Kirilov, Daan Kolkman and Bert-Jan Butijn. ***Eindhoven University of Technology, Netherlands.*** **The Elementary Scenario Component Metric for Summarization Evaluation.**
16:20 - 16:30	**Break**
16:30– 17:30	*Oral Session 7: Language Model, Security and NLP tools* **Chair: Prof. Hend Alkhalifa**
16:30 – 16:45	Ahmet Yavuz Uluslu and Gerold Schneider. ***University of Zurich, Switzerland.*** **Scaling Native Language Identification with Transformer Adapters.**
16:50 – 17:05	Ahnaf Mozib Samin, Behrooz Nikandish and Jingyan Chen. ***University of Groningen, Netherlands.*** **Arguments to Key Points Mapping with Prompt-based Learning.**
17:10 – 17:25	Farhood Farahnak and Leila Kosseim. ***Concordia University, Canada.*** **Pre-training Language Models for Surface Realization.**
17:30-17:45	*Closing session*

N.B: TIME IN GMT