

Proceedings of the Sixth
IN5550 Workshop on Neural
Natural Language Processing
(WNNLP 2024)

Andrey Kutuzov, David Samuel, Vladislav Mikhailov,
Roxana Pop and Sondre Wold (editors)

June 03, 2024

University of Oslo, Norway

Published by

Language Technology Group

Department of Informatics

University of Oslo

Preface

We are delighted to present the proceedings of the Sixth IN5550 Teaching Workshop on Neural Natural Language Processing (WNNLP 2024). Spurred by great advancements in neural approaches to NLP, this is the sixth in a series of successful annual workshops, each showcasing some of the best efforts made by MSc and PhD students completing the IN5550 class — all tackling modern research tasks related to deep learning in NLP.

The workshop received 21 submissions (by 31 authors), of which all have been accepted for publication as part of the WNNLP 2024 proceedings (this volume). This constitutes a two-fold increase in comparison to WNNLP 2023, reflecting the rising interest to neural approaches in natural language understanding and generation.

The Programme Committee has selected the paper *Fact or Fiction? Exploring Diverse Approaches to Fact Verification with Language Models*, by **Tobias Opsahl**, for the WNNLP 2024 Best Paper award. In this paper, the author applied novel methods (especially for subgraph extraction) not suggested in the track description. It allowed to outperform state of the art significantly.

The IN5550 course offered three obligatory assignments throughout the semester, with leader-boards of the best-performing submissions published for all of them. This year, the Outstanding Coursework award goes to **Vebjørn Haug Kåsene**, whose submissions were among the best (ranked as either the first or the second) in each of the three leader-boards.

Congratulations to all the award recipients (and runners-up)!

This workshop would not have been possible without the assistance of the reviewers in our Program Committee, whose careful and constructive feedback has been an important element in improving the submitted papers. All the members of the Program Committee are listed on the next page.

And of course, we extend warmest thanks to all the participants of this workshop, who spent many a sleepless nights working on the projects that made WNNLP 2024 an exciting and stimulating event!

WNNLP 2024 Committee
Oslo; June 03, 2024

Programme Committee

Aleksandar Davidov
Andreas Poole
Andrey Kutuzov
Arezo Shakeri
Dans Reinicans
David Samuel
Dennis Adelved
Eirik Eggset
Erling Midtgard
Fernando Vallecillos Ruiz
Jonas Sebulon Frydenberg
Lise Chen
Louis Give
Lucas Georges Gabriel Charpentier
Magnus Jørgenvåg
Magnus Sandem Dhelie
Maria Antonietta Bruno
Mariia Fedorova
Marthe Løken Midtgaard
Mathias Strømberg Durkis
Peder August Darre Fasting
Petter Mæhlum
Rayyan Syed
Roxana Pop
Sander Finnset Ørnes
Sindre Bjørndal-Riis
Solveig Helene Willoch
Sondre Wold
Sushant Gautam
Taras Andrushko
Timo Zaoral
Tobias Opsahl
Torkild Finne
Vebjørn Haug Kåsene
Victoria Handford
Victoria Ovedie Chruickshank Langø
Vladislav Mikhailov
Zoia Butenko

Scandinavian Language Identification track Chairs

David Samuel and Petter Mæhlum

Pretraining Tiny Language Models track Chair

Lucas Georges Gabriel Charpentier

Fact-Checking with Graph Evidence track Chairs

Roxana Pop and Sondre Wold

Artificial Text Detection track Chair

Vladislav Mikhailov

Predicting a Prompt from a Generated Text track Chair

Andrey Kutuzov

General Chair

Andrey Kutuzov

Contents

1	Scandinavian Language Identification track	1
1.1	Enhancing Language Identification Models: Insights from Annotated Sentences and Baseline Model Analysis (<i>Victoria Ovedie Chruickshank Langø, Lise Chen, Mathias Strømberg Durkis</i>)	2
1.2	Optimizing the Performance and Reducing Complexity of Simple Neural Classifiers for Scandinavian Language Identification (<i>Magnus Jørgenvåg</i>)	13
1.3	Fine-Tuning Language Identification Models for Scandinavian Languages: Insights and Comparisons (<i>Arezo Shakeri</i>)	23
1.4	Improving Scandinavian Language Identification (<i>Jonas Sebulon Frydenberg</i>)	31
1.5	Successes and Downfalls: Error Analysis of Scandinavian Language Identification (<i>Marthe Løken Midtgaard, Victoria Handford, Solveig Helene Willoch</i>) .	41
1.6	Transformers do (not) outperform FastText on Scandinavian language identification (<i>Mariia Fedorova</i>)	51
2	Pretraining Tiny Language Models track	59
2.1	An empirical study of recent optimizers in tiny language models (<i>Sander Finnset Ørnes</i>)	60
2.2	From bad babies to good language students? - An experiment of training tiny language models on ESL data (<i>Erling Midtgaard</i>)	71
2.3	Parameter Sharing for Tiny Language Models (<i>Eirik Eggset, Torkild Finne, Rayyan Syed</i>)	85
2.4	TinyLM: Back to basics (<i>Peder August Darre Fasting, Vebjørn Haug Kåsene</i>)	95
2.5	Curriculum Learning in Tiny Language Models (<i>Magnus Sandem Dhelie</i>)	103
3	Fact-Checking with Graph Evidence track	111
3.1	Enhancing Fact Verification with Hybrid Models: Integrating Graph Neural Networks and BERT Using the FactKG Dataset (<i>Aleksandar Davidov</i>)	112
3.2	FactGenius: Combining Zero-Shot Prompting and Fuzzy Relation Mining to Improve Fact Verification with Knowledge Graphs (<i>Sushant Gautam</i>)	121
3.3	Fact or Fiction? Exploring Diverse Approaches to Fact Verification with Language Models (<i>Tobias Opsahl</i>)	131
4	Artificial Text Detection track	139
4.1	Cutting Down to Size: Efficient AI for Artificial Text Detection (<i>Fernando Vallecillos Ruiz</i>)	140

4.2	Impact of Domain Variability in Training Data for Monolingual Artificial Text Detection (<i>Dans Reinicans</i>)	149
4.3	Overfitting in Artificial Text Detection: Expanding on the SemEval-2024 Workshop Task 8 (<i>Sindre Bjørndal-Riis</i>)	155
4.4	Using Statistical Features to Improve the Performance of Large Language Models in Artificial Text Detection (<i>Andreas Poole</i>)	163
4.5	Zero-shot Artificial Text Detection Using Instruct Fine-Tuned LLMs (<i>Zoia Butenko, Taras Andrushko</i>)	171
5	Predicting a Prompt from a Generated Text track	179
5.1	Exploring Prompt Prediction: Methods, Performance, and Challenges (<i>Dennis Adelved</i>)	180
5.2	Uncovering Hidden Intentions: Exploring Prompt Recovery for Deeper Insights into Generated Texts (<i>Louis Give, Timo Zaoral, Maria Antonietta Bruno</i>)	205

1 Scandinavian Language Identification track

Enhancing Language Identification Models: Insights from Annotated Sentences and Baseline Model Analysis

Victoria Ovedie Chruickshank Langø

Institute of Informatics
University of Oslo
victocla@uio.no

Lise Chen

Institute of Informatics
University of Oslo
liseche@uio.no

Mathias Strømberg Durkis

Institute of Informatics
University of Oslo
mathisdu@uio.no

Abstract

This paper addresses a common issue in language identification models: their reduced accuracy in predicting languages in short sentences. Using a multilabel dataset with Norwegian Bokmål, Norwegian Nynorsk, Swedish, and Danish, we evaluated the performance of three well-known language identification models, namely OpenLID, FastText, and NLLB (No Language Left Behind). Our analysis reveals that, although the models generally perform well, they struggle with accurately predicting languages in shorter sentences. We identify this gap and propose the development of models optimized for short sentences. This paper details our evaluation methodology, findings, and the framework for enhancing prediction accuracy in minimal text scenarios.

1 Introduction

Language identification (LID) involves determining the language of spoken or written content without prior knowledge of the speaker or writer's identity (Muthusamy et al., 1994, 33). Humans excel at identifying languages they know or noting similarities to familiar languages within moments of exposure. Similarly, effective LID systems exploit these distinctive linguistic traits to accurately identify languages, a task that remains challenging, especially in multilingual and noisy environments.

To evaluate the effectiveness of existing language identification models, we selected three LID models to serve as baselines and utilized a multi-label evaluation script on a dataset consisting of 6,976 sentences, each labeled with one or more languages. After processing the sentences through the models, we manually annotated each of the 2,336 incorrectly predicted sentences as either "wrong language" or "ambiguous." This detailed annotation process helped us gain a deeper

understanding of the specific types of sentences the models were struggling with.

Our evaluation highlighted the models' particular struggles with shorter sentences, leading us to analyze each misprediction in detail. These insights underscored the need for enhanced model performance on short sentences, prompting our research to develop more precise models. This paper addresses this gap through the following contributions:

- Evaluation of the performance of existing language identification models on a multilabeled dataset.
- Identification and analysis of specific challenges these models face with shorter sentences.
- Annotation and examination of inaccuracies in predictions to understand their causes.
- Proposal and outline of the development of new models to improve accuracy in language identification for short sentences.

2 Background and Related Work

This section provides an overview of the baseline models used in language identification tasks, specifically focusing on three baseline models: OpenLID, FastText, and NLLB. While our primary contribution is an error analysis of language identification models, we acknowledge the foundational work that has informed the development and evaluation of language identification systems.

Baseline Models:

- **OpenLID:** As part of efforts to develop robust and scalable systems for identifying spoken languages, OpenLID has been trained using an open dataset provided in their publication (Burchell et al., 2023). This dataset

also served to train a FastText-based language identification model, where embeddings generated from FastText are utilized as inputs to a multiclass linear classifier. Notably, OpenLID employs the same hyperparameters as those used in the NLLB model, ensuring consistency and comparability between the models (Burchell et al., 2023; Team et al., 2022).

- **FastText:** A text classification model developed by Facebook AI, FastText excels in handling large volumes of text efficiently, using bag-of-words and n-gram features to predict the language of textual content (Joulin et al., 2016).
- **NLLB:** An ambitious model aiming to provide translation capabilities across a wide array of languages, including low-resource languages, emphasizing its utility in broadening the applicability of language technologies to under-served languages (Team et al., 2022).

3 Methodology

3.1 Evaluation Datasets

The main evaluation dataset is derived from the test splits of the Universal Dependencies (UD) (Universal Dependencies, 2024) treebanks for Norwegian Bokmål, Norwegian Nynorsk, Swedish, and Danish. To enhance the evaluation, we used a dataset with additional labels for sentences that could belong to multiple languages (Samuel, 2024). For instance, a sentence like “Hvordan går det?” could be both Bokmål and Danish, but not Nynorsk or Swedish. This manually filtered evaluation dataset ensures comprehensive testing of the models’ ability to distinguish between closely related languages.

3.2 Annotation Process

Although our models do not predict multiple labels, we utilized this multilabel dataset to better understand the nature of the mispredictions. Initially, we aimed to see the types of errors the models made and discovered that some sentences were ambiguous. This prompted us to manually annotate each misprediction, providing deeper insights into the specific types of errors the models were making (Langø et al., 2024). For instance, the phrase “Aggresjon og alvor” is labeled as ‘nn’ in the dataset, yet a model’s prediction of ‘nb’ is also plausible given the ambiguity in language usage. The text

samples that were labeled “Error” were taken out of our analysis.

For the classification of the languages, as our team consists of three fluent Norwegian speakers with knowledge about the other Scandinavian languages, we used our previous knowledge, and supplemented with dictionaries. For Norwegian Bokmål and Norwegian Nynorsk, we employed two sources: orbokene.no (Språkrådet and University of Bergen, 2024) and the LEXIN dictionary (Oslo Metropolitan University, 2024). For Danish, we used Den Danske Ordbog (Det Danske Sprog- og Litteraturselskab, 2024), in addition to Cambridge Dictionary (Cambridge University Press, 2023) for comparisons between Bokmål and Danish. For Swedish, we used Svenska Akademiens Ordböcker (Svenska Akademien, 2024). Notably, Swedish was distinguished more readily from the other languages, a trend that is consistent across the prediction matrices (Tables 1, 2, and 3); all three models accurately predicted Swedish with high reliability.

This annotation process was designed not just to assess the overall performance of the models, but to gain deeper insights into the specific errors occurring in language identification. NLLB incorrectly predicted 390 samples, OpenLID 529 samples, and FastText 1,417 samples, totaling 2,336 misclassified samples.

The process aimed to provide a granular understanding of the predictive challenges faced by the models, enhancing our interpretation of their performance.

4 Results and Analysis From Annotation

4.1 Baseline Model Performance

- **OpenLID:** Achieved an F1-score of 92.39%. Common misclassifications were between Norwegian Bokmål and Norwegian Nynorsk.
- **NLLB:** Achieved an F1-score of 93.21%. Similar misclassification patterns to OpenLID but with fewer errors.
- **FastText:** Achieved an F1-score of 76.74%. Showed significant misclassification issues overall, especially when compared to OpenLID and NLLB.

4.2 Misclassification Analysis

The misclassification matrices for OpenLID, NLLB, and FastText reveal insights into their performance. All models struggled particularly with

Norwegian Bokmål and Norwegian Nynorsk, reflecting the linguistic similarities between these languages. The FastText model also had difficulty distinguishing between Danish and Norwegian Bokmål.

Table 1: OpenLID Misclassification Matrix

Target \ Predicted	nb	nn	da	sv	other
nb	-	78	63	16	51
nn	146	-	32	14	17
da	35	1	-	3	6
sv	2	1	2	-	4
other	13	20	14	15	-

Table 2: NLLB Misclassification Matrix

Target \ Predicted	nb	nn	da	sv	other
nb	-	26	40	11	62
nn	60	-	12	11	60
da	5	1	-	1	47
sv	2	2	2	-	28
other	49	29	32	26	-

Table 3: FastText Misclassification Matrix

Target \ Predicted	nb	nn	da	sv	other
nb	-	26	304	72	112
nn	574	-	115	61	106
da	29	1	-	12	54
sv	9	1	2	-	42
other	27	3	27	6	-

4.2.1 OpenLID Model Analysis

OpenLID frequently misclassified Norwegian Bokmål as Norwegian Nynorsk, Danish and 'Other' languages, indicating confusion between closely related Scandinavian languages and broader categories.

4.2.2 NLLB Model Analysis

NLLB showed improved predictions compared to OpenLID, with generally fewer misclassifications. However, it struggled with the 'Other' category.

4.2.3 FastText Model Analysis

FastText exhibited significant misclassification issues, especially between Norwegian Bokmål and Norwegian Nynorsk, as well as Danish and Norwegian Bokmål.

4.2.4 General Observations

All models struggled with closely related languages such as Norwegian Bokmål, Nynorsk, and Danish, yet they all performed well in predicting Swedish text samples. NLLB generally outperformed the

others, suggesting better model architecture or training methodologies.

4.3 Detailed Error Analysis

From our annotation, we discovered that OpenLID has a significantly higher number of ambiguous predictions compared to NLLB and FastText, as shown in Figure 1. This suggests that OpenLID's accuracy might be underestimated when evaluated against standard evaluation metrics, as many of its errors are due to the inherent ambiguity of the sentences rather than outright mispredictions.

4.3.1 Mean Length Analysis

To further understand the challenges faced by the models, we analyzed the mean length of text samples for wrong predictions and ambiguous predictions. The mean length of sentences in the overall evaluation dataset is 13.30.

The following tables summarize the mean lengths for each model and prediction type and the number of mispredictions by sentence length range:

Table 4: Mean Length of Text Samples for Wrong and Ambiguous Predictions

Model	Prediction Type	Mean Length	Amount
OpenLID	Wrong prediction	6.84	345
OpenLID	Ambiguous	4.97	168
NLLB	Wrong prediction	6.67	302
NLLB	Ambiguous	4.01	69
FastText	Wrong prediction	12.19	1347
FastText	Ambiguous	6.98	64

Table 5: Number of Mispredictions by Sentence Length Range

Sentence Length	OpenLID	NLLB	FastText
1-5	303	222	377
6-10	129	91	370
11-15	54	43	272
16-20	15	11	189
21-	13	9	203

As depicted in Table 5, all models struggle with shorter sentences, with the highest number of mispredictions occurring in the 1-5 word range. FastText shows the most significant difficulty across all sentence lengths, particularly in the 6-10 and 21-word ranges. OpenLID and NLLB, while performing better overall, still show a notable number of mispredictions for shorter sentences.

These results indicate that improving model performance for shorter sentences is crucial for

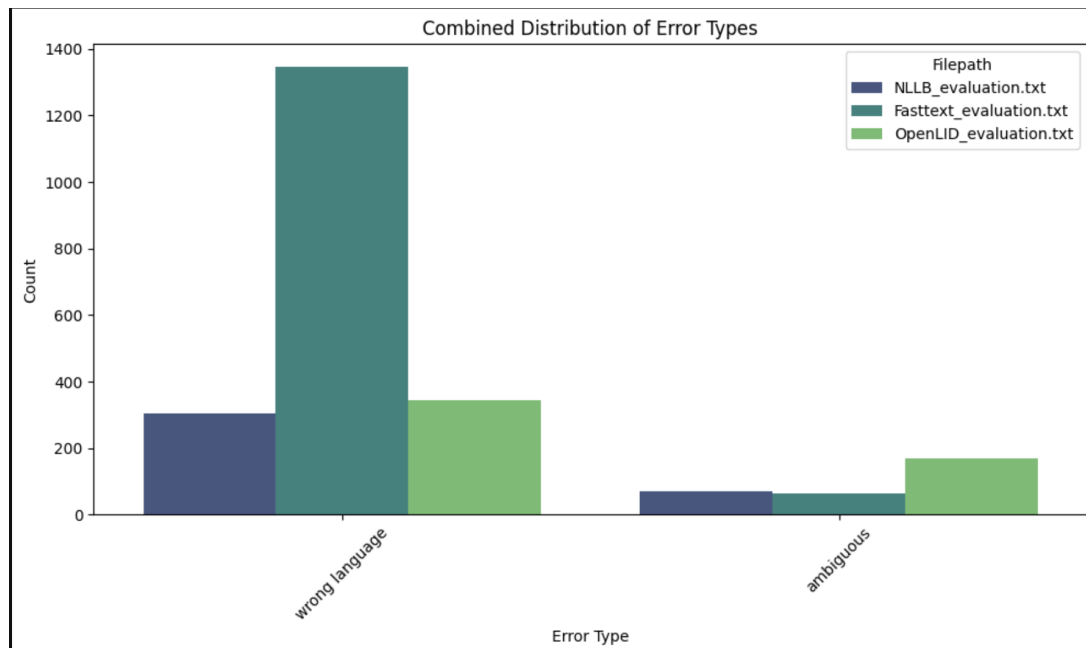


Figure 1: Combined Distribution of Error Types

enhancing overall language identification accuracy. While FastText has a high number of mispredictions across all lengths, OpenLID and NLLB show that even higher-performing models face challenges with minimal text inputs.

These findings highlight some challenges in language identification models, particularly with shorter sentences. Linguistic similarities between languages, such as Norwegian Bokmål and Nynorsk, make it difficult for models to distinguish between them. Shorter sentences provide limited context, which reduces the amount of information available for accurate language identification. Additionally, the architecture of the models plays a role; models like FastText, which are less sophisticated than OpenLID and NLLB, struggle more with these tasks. Finally, ambiguity in language use, where sentences or phrases can belong to multiple languages, further complicates the identification process. We found several examples of sentences that contained only one word or a company name originating from one of the Scandinavian languages but with a more international name. Examples include "NTB", "Scanpix", "Aschehoug", and even simple words like "Ja" proved challenging for some models to classify accurately. These factors collectively contribute to the models' difficulties with short sentences, emphasizing the need for improved model architectures and more nuanced training data to enhance performance in these scenarios.

5 Hypothesis and Future Experiments

Our analysis of the misclassification patterns and the influence of text length on prediction accuracy has led us to hypothesize that training models on datasets filtered for shorter sentences might improve their performance on minimal text inputs. Specifically, we propose to:

- Train OpenLID with the same hyperparameters as the original (Burchell et al., 2023) on more Scandinavian data, using both the gold dataset and a version filtered for shorter sentences (under 6 words).
- Fine-tune XLM-R ('FacebookAI/xlm-roberta-base') from Hugging Face (Conneau et al., 2019) on additional Scandinavian data, utilizing both the gold dataset and the filtered dataset. Integrate the predictions from both the fine-tuned XLM-R model and OpenLID (Burchell et al., 2023). The combined predictions are then used to determine the final language classification.
- Fine-tune mBERT ('google-bert/bert-base-multilingual-cased' from Hugging Face (Devlin et al., 2018)) and XLM-R ('FacebookAI/xlm-roberta-base' from Hugging Face (Conneau et al., 2019)) on more Scandinavian data, using both the gold dataset and the filtered dataset.

5.1 Methodology for Testing the Hypothesis

5.1.1 Motivation

OpenLID

We chose to experiment with OpenLID, despite it being a baseline model, mainly due to its strong performance in language identification, particularly with Scandinavian languages. Additionally, it predicted a higher number of ambiguous words compared to the other models, suggesting that the accuracy metrics could have been the highest, instead of NLLB. Our goal was to see if adding both filtered and unfiltered datasets could further improve its accuracy, especially for short sentences. By testing different dataset configurations, we aimed to enhance OpenLID’s precision and better understand its capabilities with minimal text inputs.

OpenLID With XLM-R

This approach aims to leverage OpenLID’s robustness and XLM-R’s contextual understanding to enhance overall accuracy. Using OpenLID as the base model, we fine-tune XLM-R on the datasets to provide more reliable predictions in areas where OpenLID alone is insufficient.

mBERT and XLM-R

mBERT and XLM-R are well-known multilingual transformer models, with XLM-R intended as an improvement over mBERT (Conneau et al., 2019). By fine-tuning these models, we hope to see how using more up-to-date transformer-based language models with existing multilingual capabilities perform with both unfiltered and filtered datasets.

We selected these models because we wanted to cover a spectrum of approaches from simple to more complex, allowing us to identify the specific advantages and limitations of each. This can help in understanding how different models respond to the challenges posed by short sentences and ambiguous language use, which can aid in the development of more effective language identification systems.

5.1.2 Classification Architecture of OpenLID With XLM-R

The model applies a softmax to the five predictions with the highest probability values from both the fine-tuned XLM-R and the base OpenLID model. We then apply weights of 0.7 to OpenLID’s predictions and 0.3 to XLM-R’s predictions, respectively, to combine them effectively. After weight-

ing the predictions, we aggregate them to produce a final prediction for each input sentence. This final prediction is determined by the highest combined weighted score, ensuring that both models contribute to the decision while prioritizing predictions from OpenLID.

5.1.3 Datasets

We will use the following datasets for training and evaluation:

- **Gold Dataset:** The gold training/validation data from Universal Dependencies treebanks for Norwegian Bokmål, Norwegian Nynorsk, Swedish, and Danish.
- **Filtered Dataset:** A subset of the gold dataset consisting only of sentences with 6 words or fewer, created by filtering the original dataset. The approach using OpenLID together with XLM-R uses a filtered gold dataset without quotes (sentences containing quotation marks or otherwise signifying quotation with phrases like "she said" or similar) and sentences longer than 6 words.
- **Silver Dataset:** A balanced dataset created using machine translation to decouple semantics from language, ensuring coverage of various domains.

5.1.4 Training and Evaluation

For training, we used the gold training/validation data from the UD treebanks. Additionally, we used the silver training data by using machine translation to decouple semantics from language. This approach ensures that the dataset is balanced in terms of sentence count and domain coverage for each language. The silver dataset addresses potential biases in the gold dataset and helps improve the robustness of the models.

- **OpenLID:** Trained with the same hyperparameters as the original, on both the gold dataset and the filtered dataset.
- **mBERT:** Fine-tuned on the gold dataset and the filtered dataset.
- **XLM-R:** Fine-tuned on the gold dataset and the filtered dataset.
- **OpenLID and XLM-R:** Fine-tuned on the gold dataset, the filtered dataset, and the silver dataset.

5.1.5 Expected Outcomes

From an evaluation of both mBERT and XLM-R without fine-tuning, the performance metrics were around 30-40% for all metrics. We expect that training models on the filtered dataset will result in:

- Improved accuracy and F1 scores for shorter sentences.
- Increased mean length of mispredictions, due to an increased number of correct classifications on the short sentences.
- A shift in the distribution of error types, with fewer wrong predictions and more ambiguous predictions, reflecting better handling of short, ambiguous sentences.

6 Findings From Experiments

6.1 OpenLID

Among the OpenLID models, the best-performing one was trained on the gold dataset filtered for shorter sentences. This model achieved the highest accuracy and F1 scores, indicating improved performance on minimal text inputs. Specifically, the analysis of mispredictions revealed that for this model, the mean sentence length for "Wrong Language" predictions was 8.36, with a total count of 354, while the mean sentence length for "Ambiguous" predictions was 4.10, with a total count of 51.

Compared to the previous model, the new OpenLID model's "Wrong Language" predictions had a slightly higher mean sentence length of 8.36, up from 6.84 with a similar total count of 354 from the previous 345. For "Ambiguous" predictions, the new model had a lower mean sentence length of 4.10 from 4.97 and a significantly reduced total count of 51 from the previous 168, indicating better handling of ambiguous cases. These results suggest that the new OpenLID model, trained on shorter sentences, can maintain high accuracy while effectively managing ambiguity in language identification.

6.2 OpenLID With a Fine-tuned XLM-R

In our experiments with OpenLID and XLM-R, we tested four different configurations. The results are detailed in Table 7. The configurations included training on the gold dataset, the gold dataset filtered, the gold and silver datasets, and the gold and

silver datasets filtered (with some adjustments, see 5.1.3). Among these, the best-performing model was trained on the filtered gold dataset. This model achieved the highest accuracy and F1 score, indicating improved performance on minimal text inputs. Specifically, the analysis of mispredictions revealed that for this model, the mean sentence length for "Wrong Language" predictions was 8.51, with a total count of 197, while the mean sentence length for "Ambiguous" predictions was 8.46, with a total count of 220. Compared to the previous model, the new model's "Wrong Language" predictions had a higher mean sentence length, increasing from 6.84 to 8.51, and a lower total count, decreasing from 345 to 197. For "Ambiguous" predictions, the new model had a higher mean sentence length of 8.46 and a higher ambiguous count, increasing from 168 to 220. The results suggest that filtering the gold dataset improves the overall quality of the training data, leading to better performance metrics such as accuracy and F1 scores. This is in accordance with the expected results. The error analysis indicates that the model is now more effective at handling shorter sentences and making fewer "Wrong Language" predictions.

6.3 Fine-tuned mBERT and XLM-R

The fine-tuning of mBERT and XLM-R gave us the best performance metrics out of all our approaches (see Table 7). mBERT-gold provided the best performance and shortest sentences of all the models we trained (97.28% loose accuracy, 96.91% loose macro F1, 96.24% loose macro MCC, sentence total mean length of 6.48, see Table 7), while XLM-R-gold-short produced the longest sentences (8.79, see Table 7) of all the models we trained, while still achieving adequate performance. This shows that XLM-R handles short sentences efficiently when given a filtered dataset. In addition, this simple "fine-tuning a multi-lingual transformer model" approach used the least GPU resources (see Table 8).

The fine-tuning of the mBERT and XLM-R models achieved better performance than our baseline models. However, since we were not able to fine-tune them in a way that gave us better performance when training on shorter sentences, the expected outcomes were not entirely confirmed.

6.4 Ambiguities of sentence length

It is perhaps less probable that long sentences are ambiguous. They may contain more unique markers of language, making it logical that well-

Table 6: Number of Mispredictions by Sentence Length Range with filtered gold dataset an % difference

Sentence Length	OpenLID	OpenLID XLM-R	mBERT and XLM-R
1-5	194	189	105
6-10	108	106	49
11-15	58	65	21
16-20	27	34	8
21-	19	23	3

performing models are not as good with shorter sentences because these are more likely to be ambiguous. The large portion of ambiguous sentences among the wrong predictions of the best-performing mBERT model may support this (59 out of 186 valid records being ambiguous, see 10). We also achieved the longest sentences of all the models with adequate performance with XLM-R gold short. However, it performs worse than the same model trained on more data (XLM-R gold), indicating that the reason for the increase in mean length is because it is a worse model, and the mean length of the wrong predictions will then approximate the mean length of the evaluated dataset.

6.5 Overall Findings

Our experiments provided several insights into the performance and optimization of language identification models, particularly in handling short sentences. The key findings from our analysis are as follows:

6.5.1 Effectiveness of OpenLID and Fine-Tuned Models

Despite being a baseline model, OpenLID showed significant improvements when trained on a filtered dataset of shorter sentences, achieving the highest accuracy and F1 scores among its variants. The analysis revealed that OpenLID’s simpler architecture benefits from targeted data subsets, effectively managing ambiguity and reducing mispredictions. Fine-tuned models, particularly mBERT and XLM-R, demonstrated superior performance overall. mBERT, trained on the gold dataset, was good at handling short sentences, while XLM-R, trained on the filtered dataset, displayed the longest mean sentence length for mispredictions. This suggests that while complex models like mBERT and XLM-R can capture intricate patterns in the data, OpenLID’s simpler structure can achieve high performance with optimized training data.

6.5.2 Impact on Filtered Datasets

Training on filtered datasets containing shorter sentences did not consistently enhance performance

across all models. For more complex but well-performing transformer models, having a larger and more comprehensive dataset was more beneficial. This might suggest the need to have enough data to reach the point of diminishing returns, but also that high-performing models might be sensitive to the reduced context in short sentences. The effectiveness of filtered datasets may depend on the model architecture and its capacity to learn from limited data.

6.5.3 Correlation Between Mean Sentence Length and Performance

The assumption that an increase in mean length would reflect better classification of short sentences was not conclusively supported. Our best-performing model the mBERT gold model had the shortest sentences of all the models we trained, but it also performs better overall, and a good portion of the misclassified sentences are actually correct (see 7). Considering that this approach gave us the best performance, we might ask ourselves if it is a reasonable goal to aim for a higher mean length of predictions when training on shorter sentences. This suggests that there is a more complex relationship than described in this paper between mean sentence length and performance. As Table 6 shows, there were still a significant number of mispredictions among shorter sentences.

6.6 Limitations

- **Training on Combined Datasets:** We did not train the mBERT and XLM-R models on a combined dataset of both gold and silver data. Training on both datasets might have provided more comprehensive coverage and improved the models’ robustness.
- **Multi-Label Classification:** Ideally, our models would support multi-label classification, as a portion of the errors in the current models are due to ambiguous sentences that can belong to multiple languages. Implementing multi-label classification could reduce the number of misclassifications.

Table 7: Loose Metrics on test.jsonl

Model	Loose Accuracy (%)	Accuracy (%)	Loose F1 (%)	Macro	Loose MCC (%)	Macro	Sentence Total Mean Length
FastText	79.69		76.74		75.38		11.65
NLLB	94.41		93.21		91.90		6.14
OpenLID	93.51		92.39		91.00		7.48
OpenLID Variants							
OpenLID with gold short	94.01		93.02		91.72		7.63
OpenLID with gold	93.82		92.76		91.44		7.54
OpenLID with gold and silver short	93.84		92.74		91.39		7.73
OpenLID with gold and silver	39.74		32.61		34.92		14.41
mBERT Variants							
mbert-gold	97.28		96.91		96.24		6.48
mbert-gold-short	96.07		95.77		94.87		7.86
XLM-R Variants							
xlm-r-gold	96.93		96.72		95.98		6.88
xlm-r-gold-short	96.33		96.05		95.20		8.79
OpenLID with XLM-R Variants							
OpenLID_xlm-roberta-base_gold short	94.31		93.60		92.32		7.72
OpenLID_xlm-roberta-base_gold	94.27		93.52		92.24		7.71
OpenLID_xlm-roberta-base_gold_silver	94.02		93.14		91.82		8.20
OpenLID_xlm-roberta-base_gold_silver short	94.19		93.27		91.97		7.70

Note: "short" refers to filtered data with sentences of 6 words or fewer. "Sentence Total Mean Length" is the mean length of the wrongly predicted sentences on test.jsonl.

- Dataset Quality:** There are some errors in the dataset used for evaluation. These inaccuracies can affect the training process and the final evaluation metrics of the models. Ensuring higher quality and more accurately labeled data could improve model performance.
- Short Sentence Focus:** While our primary focus was on improving performance for shorter sentences, this emphasis might have led to less optimal performance for longer sentences. A balanced approach that ensures good performance across all sentence lengths is needed, especially because one ideally wants to have good LID overall, not just on short sentences.
- Long Mean Sentence Length Focus:** Focusing on the mean sentence length for wrongly classified predictions being long may not be the best marker for a model being better at classifying short sentences, and if it is, it might be highly architecture dependent. Because longer sentences are more likely to contain unique markers for language, the mean sentence length for mispredictions should perhaps even be low. We did achieve better performance together with longer mean sentence length for wrong predictions for some of the models, but our best-performing models had a short mean sentence length. Perhaps an approach that focuses on good performance across all sentence lengths is more useful.
- Ambiguity Handling:** Although we categorized ambiguous predictions, further work is needed to develop methods for better handling and resolving ambiguities in language identification. For example, during annotation, we observed that company names like 'Aschehoug' were often predicted as 'Other.' Despite 'Aschehoug' being a Norwegian company, the name itself does not inherently appear Norwegian, making it understandable why the model might classify it as a non-Scandinavian language.
- Scandinavian Language Focus:** Our research was limited to Scandinavian languages. The findings and proposed models may not generalize well to other language families with different linguistic characteristics.

7 Conclusion

Our study addressed the challenge of predicting languages in short sentences using OpenLID, FastText, and NLLB models. While these models generally performed well, OpenLID and NLLB struggled with short sentences, and FastText did not perform very well overall. By filtering datasets for shorter sentences, we aimed to improve this aspect.

OpenLID showed significant improvement with filtered datasets, achieving both higher accuracy and F1 scores and a higher mean sentence length than its baseline OpenLID, but the model was not capture more context of shorter sentences based on a filtered dataset.

Fine-tuned models, especially mBERT and XLM-R, demonstrated superior performance. XLM-R handled short sentences effectively, while mBERT needed unfiltered datasets to get the best results.

Our findings indicate that while training on shorter sentences can enhance some models, the relationship between mean sentence length and classification accuracy is complex.

Future work should focus on developing models optimized for various sentence lengths and exploring the balance between comprehensive and targeted datasets to enhance language identification performance. Additionally, investigating why OpenLID and XLM-R seemingly improved in predicting shorter sentences, while mBERT did not, could provide valuable insights and potentially lead to the development of a superior language identification model capable of accurately predicting sentences with minimal context.

References

- Laurie Burchell, Alexandra Birch, Nikolay Bogoychev, and Kenneth Heafield. 2023. [An open dataset and model for language identification](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 865–879, Toronto, Canada. Association for Computational Linguistics.
- Cambridge University Press. 2023. [Cambridge dictionary](#). Accessed: 2024-05-10.
- 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](#). *CoRR*, abs/1911.02116.
- Det Danske Sprog- og Litteraturselskab. 2024. [Den danske ordbog: Moderne dansk sprog](#). Accessed: 2024-05-16.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, arXiv:1810.04805.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. [Bag of tricks for efficient text classification](#). *Preprint*, arXiv:1607.01759.
- Victoria Langø, Lise Chen, and Mathias Strømberg Durkis. 2024. [Annotated mispredictions](#). GitHub. Accessed: 2024-05-21.
- Y.K. Muthusamy, E. Barnard, and R.A. Cole. 1994. [Reviewing automatic language identification](#). *IEEE Signal Processing Magazine*, 11(4):33–41.
- Oslo Metropolitan University. 2024. [Lexin dictionary](#). Accessed: 2024-05-10.
- David Samuel. 2024. [Scandinavian language identification dataset](#). GitHub. Accessed: 2024-05-10.
- Språkrådet and University of Bergen. 2024. [ordbøkene.no: Bokmålsordboka og nynorskordboka](#). Accessed: 2024-05-16.
- Svenska Akademien. 2024. [Svenska akademiens ordböcker](#). Accessed: 2024-05-16.
- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Hefernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. [No language left behind: Scaling human-centered machine translation](#). *Preprint*, arXiv:2207.04672.
- Universal Dependencies. 2024. [Universal dependencies](#). Accessed: 2024-05-10.

A Appendix

Table 8: Overview of models

Model Name	Training Set	Validation Set	GPU	Training Time
OpenLID	lid201 and gold-train	*	NVIDIA A100	65 minutes and 6 seconds
OpenLID	lid201 and gold-train short	*	NVIDIA A100	66 minutes and 7 seconds
OpenLID	lid201, gold-train, silver-train short	*	NVIDIA A100	65 minutes and 4 seconds
OpenLID	lid201, gold-train, silver-train	*	NVIDIA A100	65 minutes and 6 seconds
mbert-gold	gold_train	gold_dev	NVIDIA GeForce RTX 3090	16 minutes 41 seconds
mbert-gold-short	gold_train_short	gold_dev_short	NVIDIA GeForce RTX 3090	3 minutes 10 seconds
xlm-r-gold	gold_train	gold_dev	NVIDIA GeForce RTX 3090	15 minutes 14 seconds
xlm-r-gold-short	gold_train_short	gold_dev_short	NVIDIA GeForce RTX 3090	3 minutes 26 seconds
OpenLID_xlm-roberta-base_gold_short	gold_short	gold_dev	NVIDIA A100	5 minutes 35 seconds
OpenLID_xlm-roberta-base_gold	gold	gold_dev	NVIDIA A100	5 minutes 1 second
OpenLID_xlm-roberta-base_gold_silver_short	gold_silver_short	gold_dev	NVIDIA A100	7 minutes 1 second
OpenLID_xlm-roberta-base_gold_silver	gold_silver	gold_dev	NVIDIA A100	60 minutes 35 seconds

Table 9: Model Hyperparameters

	OpenLID	mbert-gold	OpenLID_xlm-r_gold_short
Learning Rate	0.8	5e-5	0.00001
Batch Size	-	16	16
Epochs	2	3	5
Optimizer/Loss	Softmax	AdamW	AdamW
Scheduler	-	Linear	Linear
Warm-up Steps	-	0	500
Scheduler Training Steps	-	8715 (all steps)	-
Embedding Dimension	256	-	-
Min Word Occurrences	1000	-	-
Char N-grams	2-5	-	-
Word N-grams	1	-	-
Bucket Size	1,000,000	-	-
Threads	68	-	-

Table 10: Misclassified Sentences Statistics for Evaluation on test.jsonl

Model	Total Records	Valid Records	Wrong Language		Ambiguous	
			Mean Length	Total Count	Mean Length	Total Count
mbert-gold	211	186	6.07	127	6.81	59
OpenLID (Gold short)	423	406	8.36	354	4.10	51
OpenLID_xlm-r_gold_short	432	417	8.51	197	8.46	220

Note: Valid records are sentences where the label is correct for the given sentence. Invalid records are sentences where the label does not correspond to the actual language of the sentence.

Optimizing the Performance and Reducing Complexity of Simple Neural Classifiers for Scandinavian Language Identification

Magnus Jørgenvåg
magnjorg@uio.no

Abstract

In practice, it is beneficial for language identification models to have fast inference and low complexity to speed up data collection. Nevertheless, current low-complexity models fail when it comes to classifying languages that are closely related to each other, like Norwegian, Danish, and Swedish. Fine-tuning a Transformer model may lead to a significant improvement in performance, but at the expense of longer inference times and increased complexity. In this report, we explore techniques to shrink the performance gap between complex Transformer models and simpler neural classifiers in Scandinavian language identification. Our best-performing approach shows that the performance of simple neural classifiers can be improved by distilling knowledge from a BERT-like teacher model. By distilling knowledge from a fine-tuned Scandi-NLI to a BiGRU model, we can improve F1-score from 0.930 to 0.950 on our validation set, while obtaining a model size that is roughly one-seventh of the best-performing baseline.

1 Introduction

In the context of the recent innovations in Language Technology, a misconception is that language identification (LID) is a simple task. This is true in many cases where the languages considered are largely different from each other, which can be efficiently solved with simple machine-learning techniques. This is the case for successful models like fastText (Joulin et al., 2016) and NLLB (Joakim Nivre and Tyers, 2017), which are all trained on a large set of different languages.

However, this problem becomes more complex when considering similar languages, like Norwegian, Swedish, and Danish. Such languages have many overlapping features in terms of grammar, vocabulary, and syntax which could be difficult for automated systems to distinguish. Moreover,

many of today’s methods are too generalized to recognize more intricate relationships between linguistically similar languages.

To address this issue, researchers have been able to improve accuracy in more complicated language identification problems by fine-tuning Transformer-based models (Tonja et al., 2022). Even though these methods can achieve state-of-the-art performance in these tasks, the challenge of long inference time and high model complexity remains. These limitations result in these models being useless for most practical applications where a language is required to be identified quickly.

To bridge the gap between poor performance and high complexity, we explore three methods of improving the performance of simple neural classifiers in Scandinavian LID¹: fine-tuning, data augmentation and knowledge distillation. Our first approach involves fine-tuning a simple neural classifier on a small Scandinavian dataset. For this task, we consider the fastText model, which provides efficient classification of 294 languages (Joulin et al., 2016). Secondly, we investigate if performance can be improved by increasing the size and variability of our training set by training a BiGRU. Lastly, we fine-tune *Scandi-NLI*, a BERT-based model, which achieves state-of-the-art performance in this field. Furthermore, we implement a technique to distill knowledge from this model to a simple one-layer BiGRU.

¹The written languages considered are Norwegian Bokmål, Norwegian Nynorsk, Danish and Swedish

2 Related work

2.1 Language identification

The earliest work formulating the LID task as a statistical problem was in 1965 by (Mustonen, 1965), who applied discriminant analysis to distinguish languages like English, Swedish and Finnish by a word-level. This was done by defining a list of character-based features typical for a particular language while computing the compatibility through a set of if-statements (Jauhiainen et al., 2018).

In the 70s, the first known machine-learning approaches were applied to language identification by (Nakamura, 1971) and (M. D., 1974). These initial methods were mainly based on occurrence rates of characters and words in each language, by applying character Bigrams and later Bayesian models (Jauhiainen et al., 2018). Compared to the early rule-based methods, machine learning techniques introduced the advantage of not having to define the linguistic nuances of each language, which is a tedious process even for experts.

In earlier work, this problem is mostly approached by simple neural classifiers. The fastText model (Joulin et al., 2016) from Facebook’s AI research group has been proven to be effective for a wide range of language identification tasks, despite only being composed of a single hidden layer with N-gram features as input. Additionally, several extensions like OpenLID (Burchell et al., 2023) and No Language Left Behind (NLLB) (Joakim Nivre and Tyers, 2017) have been developed to improve accuracy and address low variability in language data (201 languages for OpenLID). These models benefit by performing well on a wide range of languages, despite having low complexity and inference time. Nevertheless, as there is large variability between the languages in the training data, the model tends to ignore high-level features that are viable when distinguishing highly similar languages.

Recently, researchers have experimented with fine-tuning Transformer models containing cross-lingual dependencies for LID, like XLM-RoBERTa (Conneau et al., 2019) and BERT-multilingual (Devlin et al., 2018a). These methods have increased the performance in more complex LID tasks like spoken language identification (Nie et al.,

2022) and offensive language identification (Alavi et al., 2021). Despite this, the long inference time and large size of BERT-like models make them impractical in real-world applications that requires high speed.

2.2 Knowledge distillation

The concept of *knowledge distillation* was first introduced by (Hinton et al., 2015) in 2015 where they established a technique of compressing knowledge from a more complex *teacher* model to a simpler *student* model. There are several variations of this method, where some methods infer knowledge between intermediate layers (Romero et al., 2015), while others explore the relationship between feature maps (Yim et al., 2017). The vanilla approach is known as *response-based* knowledge distillation, where knowledge is distilled from the last output layer. In particular, this is done by minimizing the soft logits between the student and teacher during training (Gou et al., 2021).

Recently, researchers have applied this method in the context of BERT models. A successful example is provided in the DistilBERT paper (Sanh et al., 2020), where they were able to reduce the size of the model by 40% and increase inference by 60% by applying a technique called self-distillation. Furthermore, (Tang et al., 2019) proved that a single-layer BiLSTM could achieve comparable performance to a BERT model by applying the response-based technique. Moreover, they were able to reduce the number of parameters by 99%, while achieving 15-times faster inference.

3 Data

3.1 Comparing the Scandinavian languages

Scandinavian LID is a more difficult task due to the similarities between the four written languages. For instance, the languages share similar grammatical structures, having the Subject-Verb-Object word order and similar systems of gender (gun, 2024). Moreover, they have a high overlap in vocabulary, which makes them mutually intelligible to varying extents.

Moreover, Norwegian Bokmål, Norwegian Nynorsk, and Danish have highly similar orthography, while sharing the same writing system. Thus, several words are spelled identically between the languages, like the word “konge” (king) and “skole” (school). Due to this, some sentences might coincide like these:

- Jeg er glad (I am happy)
- Det er kaldt (It is cold)
- Vi ser deg (We see you)

Hence, it is more difficult to separate these languages, as some sentences would only differ by small differences in spelling or by combining words that can not be done in the other languages.

On the other hand, Swedish has a higher proportion of unique words compared to the other Scandinavian languages, while words are often spelled differently. For example, the word for “river” is spelled like “älv” in Swedish, while it is spelled like “elv” in the other three languages. Furthermore, the Swedish written language has the vowels “ä” and “ö” that not used in the other Scandinavian languages. Thus, despite sharing many similarities with the other Scandinavian languages, Swedish is easier to separate due to these qualities.

3.2 The UD dataset

The models presented in this paper are trained on the treebanks provided by Universal Dependencies (Nivre et al., 2017), which are gathered from news articles. In particular, we use all data from *no_bokmaal* (Norwegian Bokmål), *no_nynorsk* (Norwegian Nynorsk), *da_ddt* (Danish) and *sv_talbanken* (Swedish). In addition, we include a label called ‘other’, which includes data randomly sampled from the other treebanks.

Furthermore, our evaluation set is sampled from the test splits of the UD treebanks, where additional labels are included if a sentence can belong to multiple languages. For this project, a prediction is defined to be correct if it shares a label with the actual target. Considering this, the number of sentences within each language and split can be seen in **Table 1**

The data in *Section 4.4* and *Section 4.3* are prepared with subword-tokenization using the BertTokenizer

Language	Train	Valid	Eval
Bokmål	15,435	2392	1939
Danish	4376	563	565
Nynorsk	14,029	1859	1511
Swedish	4278	503	1219
Other	8805	1165	1742
<i>Sum</i>	46,923	6482	6976

Table 1: Distribution of sentences within each language in our dataset

provided by Huggingface’s transformer library (Wolf et al., 2019). Based on an analysis of the distribution of the input lengths after tokenization, we choose a maximum input length of 95 for our models, as it covers 99.5% of all sentences in the training set. Furthermore, we did not include sequence length in our hyperparameter search, as we aim to keep the complexity of our models to a minimum.

3.3 The XL-UD dataset

An issue with the above dataset is that it is relatively small. Thus, we construct a dataset that we call the *XL-UD dataset* where data is sampled from both the UD dataset and OpenLID dataset (Burchell et al., 2023).

Firstly, the dataset presented in *Section 3.2* is extended by translating all sentences into the opposite languages using a reliable machine translation model. This is done to prevent bias towards a certain language based on certain topics. To ensure a balanced dataset, 10000 additional sentences are collected from the Papluca LID dataset (Papariello, 2022) to increase the proportion of non-Scandinavian languages.

Moreover, we include the Scandinavian portions of the OpenLID dataset (Burchell et al., 2023), which accounts for 3,973,481 sentences in total. In addition to this, we include 1,000,000 sentences randomly sampled from other languages to account for non-Scandinavian languages. Moreover, the number of sentences distributed over each language and dataset split can be seen in **Table 2**.

Language	Train
Bokmål	1,113,120
Danish	1,264,301
Nynorsk	491,345
Swedish	1,258,499
Other	1,018,806
<i>Sum</i>	5,146,071

Table 2: Distribution of sentences within each language in the XL-UD dataset

4 Experimental setup

4.1 Baseline models

We establish a set of baselines that will provide a point of reference to the other models, which will be the original fastText (Joulin et al., 2016), NLLB (Joakim Nivre and Tyers, 2017) and OpenLID (Burchell et al., 2023) models. In addition, the results are compared with a random identifier, where a label is chosen at random. By utilizing facebook’s *fastText* library in Python (Mikolov et al., 2018), we run inference on all samples on the validation and evaluation sets, while measuring performance metrics like accuracy, precision, recall, F1-score and Matthews’ correlation coefficient (Chicco et al., 2021).

4.2 Fine-tuning a fastText model

In our first experiment, we explore whether the performance of simple neural classifiers in Scandinavian LID can improve by fine-tuning a baseline. In particular, we fine-tune the fastText model on the UD dataset from Section 3.2 and evaluate its performance against the baselines. To ensure optimal performance, we conduct a hyperparameter search of the most vital hyperparameters using the following grid:

- *Learning rate*: [0.01, 0.1, 0.5]
- *Number of epochs*: [5, 10, 25, 50]
- *Word N-grams*: [1, 2, 3]

After conducting the search, the best-performing hyperparameters are at *learning_rate* = 0.1, *epochs* = 50 and *word_ngrams* = 2². Moreover, the model had a total training time of 29.59 seconds, while containing 44,363,400 parameters.

²The full results of the search can be seen in Appendix A.1

val_loss	acc.	lr	w_decay
0.1321	0.9753	1.325e-05	0.2390
0.1342	0.9754	2.421e-05	0.2932
0.1344	0.9703	9.214e-06	0.2038
0.1429	0.9721	8.931e-06	0.1739
0.1493	0.9692	4.921e-05	0.1631
0.1502	0.9732	4.992e-05	0.1409
0.1504	0.9661	7.948e-06	0.1294
0.1720	0.9710	5.392e-06	0.0385

Table 3: The results on validation loss (val_loss) population-based training on the Scandi-NLI on accuracy (acc.), learning rate(lr) and weight decay (w_decay)

4.3 Fine-tuning a BERT-model

Furthermore, we fine-tune a BERT model for the same task on the UD dataset. This will serve as the teacher model in Section 4.5, as well as another baseline comparison to our simple neural classifiers. For this task, it is beneficial to choose a model that already contains cross-lingual dependencies in the Scandinavian written languages. Keeping this in mind, we choose to fine-tune the Scandi-NLI model (Institute, 2022), which achieves state-of-the-art performance in Norwegian, Swedish, and Danish.

To ensure the optimal selection of hyperparameters, we use the Ray library in Python (Moritz et al., 2017) to implement population-based training. Differently from regular grid search, population-based training exploits information from previous runs to find the best-performing hyperparameters during the search. In our search, we explore different learning rates (ranging between $5e - 6$ to $5e - 5$) and values of weight decay (from 0 to 0.3) over 3 epochs and 9 trials. The search took 14.27 hours in total, with the best-performing hyperparameters achieving minimal validation loss at *lr* = $1.325e - 05$ and *weight_decay* = 0.239. The full results of the hyperparameter search can be seen in Table 3.

The model was trained on the UD dataset by utilizing the Trainer class from the Huggingface library (Wolf et al., 2019) while using the Adam optimizer and cross-entropy loss function. Furthermore, we train the model for 6 epochs while saving checkpoints and evaluating every epoch. The minimum validation loss ended up being at *epoch* = 1 with total training time taking 7.3

Model	Accuracy		Precision		Recall		F1		MCC		Inference	
	val.	eval.	val.	eval.	val.	eval.	val.	eval.	val.	eval.	val.	eval.
<i>Random</i>	0.205	0.211	0.206	0.210	0.200	0.213	0.186	0.204	0.007	0.012	0.01s	0.01s
<i>fastText</i>	0.730	0.796	0.743	0.801	0.813	0.806	0.742	0.776	0.657	0.747	0.20s	0.23s
<i>NLLB</i>	0.931	0.944	0.918	0.937	0.947	0.946	0.930	0.941	0.907	0.928	0.64s	0.67s
<i>OpenLID</i>	0.918	0.935	0.906	0.924	0.931	0.937	0.918	0.930	0.889	0.917	0.62s	0.65s
<i>S-NLI</i>	0.976	0.988	0.982	0.988	0.978	0.987	0.980	0.988	0.967	0.984	151s	164s
<i>GRU</i>	0.934	0.939	0.936	0.936	0.937	0.940	0.936	0.938	0.910	0.922	0.12s	0.14s
<i>FT-Tuned</i>	0.946	0.946	0.949	0.947	0.947	0.947	0.948	0.948	0.927	0.936	0.52s	0.52s
<i>XL-GRU</i>	0.892	0.910	0.885	0.900	0.898	0.906	0.891	0.902	0.854	0.884	0.13s	0.14s
<i>D-GRU</i>	0.946	0.949	0.958	0.948	0.943	0.945	0.950	0.946	0.926	0.934	0.12s	0.13s

Table 4: Accuracy, precision (macro), recall (macro), F1-score (macro) and Matthews Correlation Coefficient (MCC) computed on the validation and evaluation set. The models are a random identifier, fastText, NLLB, OpenLID, Scandi-NLI, GRU, XL-GRU and Distil-GRU in that order. Bold values represent the best-performing score of the FT-Tuned, XL-GRU, and Distil-GRU models

hours on an NVIDIA RTX A2000 8GB GPU for all epochs. Moreover, total number of parameters for the Scandi-NLI model is 354M.

4.4 GRU

We train a simple BiGRU model, which is implemented in Pytorch library (Paszke et al., 2019). Firstly, the input is encoded with BERT’s tokenizer (Devlin et al., 2018b), in which words are split into subword tokens. The model architecture consists of a single embedding layer of dimension 128 together with a bi-directional GRU module, which is restricted to a single layer to limit the complexity of our model. Lastly, we include a linear layer that maps to a probability distribution over the 5 labels, which results in a model having 6,665,477 parameters. To explore how data augmentation influences performance, we train a model on the UD dataset (LSTM) and another on the XL-UD dataset (XL-LSTM).

The parameters are chosen based on experimentation, previous experiences, and a grid search. The GRU architecture was chosen, as it normally achieved the best results on the validation set across different hyperparameter settings compared to the LSTM. To stabilize training and avoid overfitting, we include Pytorch’s layer normalization and exponential learning rate scheduler (Paszke et al., 2019), where $\gamma = 0.9$. Otherwise, we perform a grid search on learning rate (LR) and weight decay with the following hyperparameter settings³:

³The grid search is only performed on the UD dataset

- *LR*: [1e-1, 6e-2, 1e-2, 6e-3, 1e-3, 6e-4, 1e-4]
- *Weight decay*: [0, 1e-5, 1e-4, 1e-3, 1e-2]

Moreover, we choose the best performing hyperparameters from the search, which is at a learning rate of $6e - 3$ and a weight decay of $1e - 4$, achieving a minimum validation loss of 0.1563⁴.

The models are trained on the same NVIDIA GPU as in Section 4.3 with the Adam optimizer together with the cross-entropy loss function from Pytorch (Paszke et al., 2019). We also introduce early stopping if validation loss does not decrease over 5 epochs. The regular LSTM stopped training after 7 epochs which took a total of 2.5 minutes, while the XL-LSTM stopped training after 10 epochs, taking a total of 9.1 hours.

4.5 Knowledge distillation

Lastly, we fine-tune a GRU model on the UD dataset, while distilling knowledge from the Scandi-NLI model fine-tuned in Section ???. In particular, we apply response-based knowledge distillation to align the predictions from the student and teacher models. This is done by modifying the loss function like in 1.

$$\mathcal{L} = \alpha \times \mathcal{L}_{CE} + (1 - \alpha) \times \mathcal{L}_{distill} \quad (1)$$

In the formula, \mathcal{L}_{CE} is the traditional cross-entropy loss, which is computed in respect to the actual labels. Furthermore, we compute the mean-squared error between the student and teacher like in 2, with

⁴Please refer to Appendix A.2 for the full results

$\mathbf{z}^{(T)}$ being the teacher’s soft logits and $\mathbf{z}^{(S)}$ being the student’s (Tang et al., 2019).

$$\mathcal{L}_{distill} = \|\mathbf{z}^{(T)} - \mathbf{z}^{(S)}\|_2^2 \quad (2)$$

We reuse the same architecture and hyperparameter settings as in Section 4.4. Furthermore, we set $\alpha = 0.1$, which makes the teacher’s logits more influential than the actual labels. The model training stopped at $epoch = 13$, which took 27.3 hours due to the long inference time of the Scandi-NLI model.

5 Results

5.1 Performance metrics

After training the models as specified in Section 4, we evaluate the models on accuracy, precision, recall, F1-score, and Matthew’s correlation coefficient (MCC). The results after computing these metrics on both the validation set and evaluation set can be seen in Table 4.

As shown in the first four rows of the table, our baseline models score differently across the evaluation metrics. As expected, the random identifier performs relatively poorly on all metrics, while the fastText model is the second-worst performing model, achieving a F1-score of 0.776 and an accuracy of 0.796 on the evaluation set. Moreover, the NLLB model performs the best across the baselines, obtaining slightly better scores than OpenLID.

Moreover, the next two rows visualize the performance of the Scandi-NLI model (S-NLI) and the regular GRU model. As expected, Scandi-NLI achieves the best scores across all metrics out of all models, while obtaining an evaluation F1-score of 0.988. Nevertheless, the inference times over the datasets are 151 and 164 seconds for the validation and evaluation sets respectively, making this approach largely inefficient. Moreover, the performance of the standard GRU model is comparable to the NLLB model, performing better on the validation set among all metrics except recall, while slightly worse on the evaluation set.

Furthermore, we evaluate the fastText-tuned (FT-tuned) model, XL-GRU and Distil-GRU (D-GRU) on the same metrics, which can be seen in the three last row of Table 4. The results show that both the fastText-tuned model and

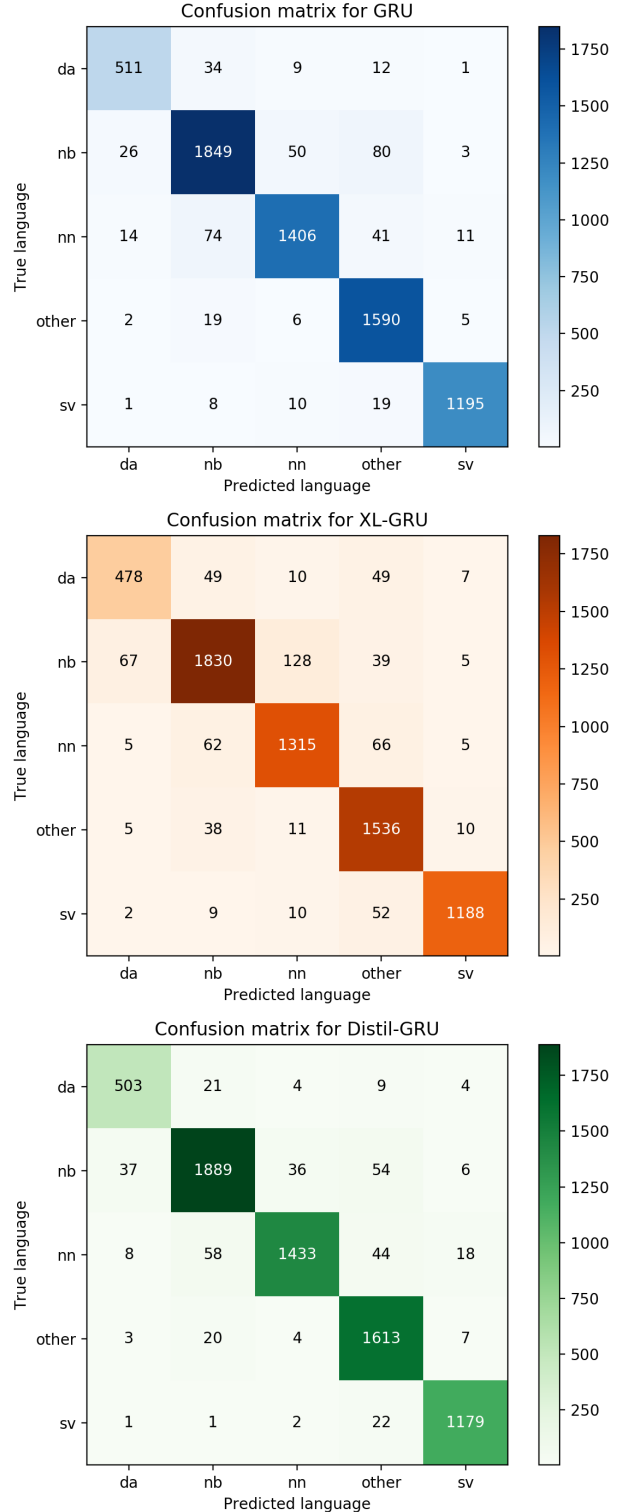


Figure 1: Confusion matrix of the regular GRU, XL-GRU and Distil-GRU calculated on our test set

Distil-GRU can achieve higher evaluation scores than all baselines except the teacher. The model performing the best among the fastText-tuned and Distil-GRU models depends on the metric,

with the Distil-LSTM model performing the best in terms of accuracy and precision, and the fastText-tuned model scoring the best scores in recall and MCC. Additionally, the Distil-GRU score is higher than the standard GRU on all metrics, thus proving the effectiveness of our knowledge distillation technique in shrinking the performance gap between Transformer models and simpler neural classifiers. At the other end, the XL-GRU model achieves worse scores than the NLLB and OpenLID baselines, which may be due to the choice of validation and evaluation sets, as discussed in *Section 6*.

5.2 Confusion matrix

Moreover, we discuss how our GRU models compare per language through confusion matrices in *Figure 1*. Analyzing these figures reveals several interesting insights:

- The Distil-GRU model has the most accurate predictions among most languages, followed by the standard GRU model.
- Interestingly, Swedish is the only language where the standard GRU model (macro F1-score of 0.976) performs better than the Distil-GRU (macro F1-score of 0.975).
- Among all models, Swedish achieves the highest per-class F1 scores⁵. This is also expected, as the Swedish alphabet has characters that are not used in the three other written languages.
- For the XL-GRU, a common mistake is to predict a sentence in Norwegian Nynorsk as Norwegian Bokmål (128 of 1474). Although less prominent, this is also a common mistake for the Distil-GRU and GRU models.
- Danish achieves the lowest per-class F1 scores among all languages⁶. This may be related to the number of Danish samples being more limited than other languages in the training data.

⁵0.976, 0.975 and 9.960 for the GRU, Distil-GRU and XL-GRU respectively

⁶0.912, 0.920 and 0.831 for the GRU, Distil-GRU, and XL-GRU respectively

6 Discussion and future work

Our results are promising, and shows that both fine-tuning and knowledge distillation are effective approaches to improve the performance of simple neural classifiers for LID of Scandinavian languages.

Nevertheless, the results of the Distil-GRU are arguably more remarkable, as model complexity is significantly reduced. This is because the size of our GRU architecture is roughly a seventh of the fastText model (6,665,477 vs. 44,363,400 parameters), while inference time is only around a fourth (0.13 vs. 0.52 seconds on the evaluation set). Thus, the Distil-GRU is the most successful in our task, as it boosts performance while limiting model complexity. Despite significantly smaller model sizes, the Distil-GRU model can achieve performance scores closer to a Transformer model than previous. Thus, we show that it is possible to boost the performance of simple neural classifiers for Scandinavian LID without compromising model complexity.

There are various directions to go in future work to further shrink the gap between Transformer models and simpler neural classifiers. For instance, we may include more diverse data in the validation and evaluation sets from more sources than only news articles. By doing this, we may obtain more reliable results for the XL-GRU model, where metric scores are likely impacted by a high bias toward the news article language. Other ways to improve performance could be to explore other knowledge distillation approaches, like feature-based or relation-based knowledge distillation, which may benefit from including context from the teacher’s intermediate layers (Gou et al., 2021). Moreover, a more extensive hyperparameter search including a larger range of settings would be beneficial. For instance, it would be helpful to include the *alpha* and *temperature* parameters in the search, which may improve knowledge distillation.

7 Conclusion

In this project, we showed that it is possible to improve the performance of simple neural classifiers for Scandinavian LID while reducing model complexity. This was done by comparing three approaches: fine-tuning a fastText model, increasing the size of the training set, and distilling knowledge from a more complex Scandi-NLI model. By doing this, we saw that our knowledge distillation technique was the most successful, as we were able to increase the F1-score from 0.930 to 0.950 on the validation set while limiting model complexity by roughly one-seventh. Thus, we show that it is not necessary to rely on more complex models like the Transformer to improve the performance of LID between linguistically similar languages.

References

2024. Nordiske språk. Store norske leksikon. Accessed on 26. May 2024 from https://snl.no/nordiske_spr
- Peyman Alavi, Pouria Nikvand, and Mehrnosh Shamsfard. 2021. [Offensive language detection with bert-based models, by customizing attention probabilities](#). *Preprint*, arXiv:2110.05133.
- Laurie Burchell, Alexandra Birch, Nikolay Bogoychev, and Kenneth Heafield. 2023. [An open dataset and model for language identification](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 865–879, Toronto, Canada. Association for Computational Linguistics.
- Davide Chicco, Niklas Töttsch, and Giuseppe Jurman. 2021. [The matthews correlation coefficient \(mcc\) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation](#). *BioData Mining*, 14(1):13.
- 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](#). *CoRR*, abs/1911.02116.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. [Knowledge distillation: A survey](#). *International Journal of Computer Vision*, 129(6):1789–1819.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. [Distilling the knowledge in a neural network](#). *Preprint*, arXiv:1503.02531.
- Alexandra Institute. 2022. ScandiNLI - natural language inference model for scandinavian languages. <https://huggingface.co/alexandrainst/scandinli-large>. Accessed: 2024-05-05.
- Tommi Jauhaainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. 2018. [Automatic language identification in texts: A survey](#). *CoRR*, abs/1804.08186.
- Filip Ginter Joakim Nivre, Daniel Zeman and Francis Tyers. 2017. Universal dependencies.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. [Bag of tricks for efficient text classification](#). *Preprint*, arXiv:1607.01759.
- Rau M. D. 1974. Language identification by statistical analysis,. Naval Postgraduate School.
- Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhresch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.
- Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, William Paul, Michael I. Jordan, and Ion Stoica. 2017. [Ray: A distributed framework for emerging AI applications](#). *CoRR*, abs/1712.05889.
- S. Mustonen. 1965. Multiple discriminant analysis in linguistic problems. page 37–44. *Statistical Methods in Linguistics*.
- Y. Nakamura. 1971. Identification of languages with short sample texts – a linguometric study. page 459–481. *Library and information science*.
- Yuting Nie, Junhong Zhao, Wei-Qiang Zhang, and Jinfeng Bai. 2022. [Bert-lid: Leveraging bert to improve spoken language identification](#). *Preprint*, arXiv:2203.00328.
- Joakim Nivre, Daniel Zeman, Filip Ginter, and Francis Tyers. 2017. [Universal Dependencies](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, Valencia, Spain. Association for Computational Linguistics.
- Luca Papariello. 2022. The papluca language identification dataset. <https://huggingface.co/papluca>. Accessed: 2024-05-05.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach 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](#). *CoRR*, abs/1912.01703.
- Julio Ariel Romero, Roberto Sanchis, and Elena Arrebola. 2015. Experimental study of event based pid controllers with different sampling strategies. application to brushless dc motor networked control system. In *2015 XXV international conference on information, communication and automation technologies (ICAT)*, pages 1–6. IEEE.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. [Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter](#). *Preprint*, arXiv:1910.01108.
- Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. [Distilling task-specific knowledge from BERT into simple neural networks](#). *CoRR*, abs/1903.12136.
- Atnafu Lambebo Tonja, Mesay Gameda Yigezu, Olga Kolesnikova, Moein Shahiki Tash, Grigori Sidorov, and Alexander Gelbuk. 2022. [Transformer-based model for word level language identification in code-mixed kannada-english texts](#). *Preprint*, arXiv:2211.14459.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. [Huggingface’s transformers: State-of-the-art natural language processing](#). *CoRR*, abs/1910.03771.
- Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. 2017. [A gift from knowledge distillation: Fast optimization, network minimization and transfer learning](#). In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7130–7138.

A Appendix

A.1 Hyperparameter search - fastText

Accuracy	lr	epochs	word_ngrams
0.9462	0.1	50	1
0.9461	0.1	50	2
0.9456	0.5	50	3
0.9454	0.1	50	3
0.9454	0.5	25	1
0.9451	0.5	10	1
0.9449	0.5	25	2
0.9448	0.5	10	3
0.9448	0.5	25	3
0.9448	0.5	50	2
0.9447	0.1	25	1
0.9445	0.5	50	1
0.9442	0.5	5	3
0.9434	0.1	25	2
0.9431	0.5	5	1
0.9428	0.1	25	3
0.9426	0.5	5	2
0.9417	0.5	10	2
0.9402	0.1	10	1
0.9391	0.1	10	2
0.9383	0.1	10	3
0.9305	0.01	50	1
0.9297	0.1	5	1
0.9285	0.1	5	2
0.9282	0.01	50	2
0.9269	0.01	50	3
0.9266	0.1	5	3
0.8624	0.01	25	1
0.8526	0.01	25	2
0.8438	0.01	25	3
0.4858	0.01	10	1
0.4653	0.01	10	2
0.4519	0.01	10	3
0.3721	0.01	5	1
0.3721	0.01	5	2
0.3721	0.01	5	3

A.2 Hyperparameter search - GRU

val_loss	acc.	epoch	lr	w_decay
0.1563	0.9416	3	0.006	0.0001
0.1581	0.9415	3	0.01	0.0001
0.1669	0.9418	3	0.006	1e-05
0.1719	0.9356	3	0.01	0.001
0.1731	0.9413	4	0.006	0.0001
0.1748	0.9415	5	0.006	0.0001
0.1750	0.9415	6	0.006	0.0001
0.1750	0.9415	7	0.006	0.0001
0.1750	0.9415	8	0.006	0.0001
0.1750	0.9415	9	0.006	0.0001
...
1.4466	0.3718	19	0.1	0.01
1.4466	0.3718	20	0.1	0.01
1.4466	0.3718	21	0.1	0.01
1.4618	0.3672	2	0.06	0
1.4802	0.3819	2	0.06	1e-05
1.4891	0.3718	2	0.1	0.01
1.5292	0.3917	2	0.1	1e-05
1.5657	0.3636	2	0.1	0.0001
1.6553	0.3190	2	0.1	0
1.7018	0.2775	2	0.06	0.0001

Fine-Tuning Language Identification Models for Scandinavian Languages: Insights and Comparisons

Arezo Shakeri

arezo.shakeri@uis.no

Abstract

Language identification forms the cornerstone of preprocessing in the analysis of large text corpora. Unfortunately, existing identification systems exhibit shortcomings, particularly in distinguishing between closely related Scandinavian languages. In this paper, we address the aforementioned challenge, focusing on the Scandinavian languages: Danish, Norwegian Bokmål and Norwegian Nynorsk, and Swedish. These languages, due to their close linguistic ties, present a unique challenge in language identification. In our study, we assessed six baseline models alongside a transfer-learning approach. We computed two versions of metrics, namely loose and strict. Among the baseline models, NLLB emerged as the top performer, achieving a strict accuracy of 91.71 and a loose accuracy of 94.41. However, our proposed transfer-learning method, utilizing the XLM-RoBERTa-base model, demonstrated improved performance metrics at the expense of runtime.

1 Introduction

Language identification, along with its associated encoding identification task, serves as an essential initial phase in numerous natural language processing endeavors (Brown, 2013). The challenge of distinguishing between closely related languages represents a significant obstacle for contemporary state-of-the-art language identification systems, despite recent efforts to expand their discriminatory capacity across a broader spectrum of languages (Zampieri et al., 2014).

Language identification serves as a fundamental preprocessing step for extensive textual datasets. For example, when training a Swedish language model, the training data is labeled as 'Swedish', and similarly, a Nynorsk machine translation model learns to generate text classified as 'Nynorsk'. Consequently, the efficacy of these models is constrained by the precision of a basic language identi-

fication tool employed in constructing the training corpus.

The current language identification models struggle to effectively differentiate among the Scandinavian languages. The aim of this paper is to assess the performance of the transfer learning approach utilizing the XLM-Roberta model for Scandinavian language identification and to evaluate the impact of incorporating synthetic data generated using a machine-translation tool. This work centers on the Scandinavian languages, particularly Danish, Norwegian Bokmål, Norwegian Nynorsk, and Swedish, due to their close linguistic relationship, presenting an intriguing challenge in language identification. Additionally, it incorporates a classification category "other" for languages outside of the mentioned languages. We assess six baseline methods, described in Section 4 and further apply a transfer-learning approach based on gold training/validation data from Universal Dependencies treebanks (Nivre et al., 2017) and a collection of machine-translated versions of gold data to generate a silver training dataset.

Finally, we evaluated both the baseline and fine-tuned models on the held-out manually-annotated test set called "Manually Filtered Evaluation Dataset" from the UD treebanks for each of the included languages, with the incorporation of supplementary labels where sentences exhibited potential belonging to multiple languages.

2 Related Work

Recent research has delved into the challenge of distinguishing between similar languages. The paper by (Zampieri et al., 2015) presents the results of the second edition of the discriminating between dimilar languages shared task. The dataset utilized in this study covered 14 languages, including an "other" category for languages that the model was not trained on. Two different test sets were

used in the evaluation. The best-performing model achieved an average accuracy of 95.54% on the original text set and 94.01% on the test set with named entities replaced by placeholders. Participants in this shared task employed a variety of classifiers and features, such as Support Vector Machine and Term Frequency-Inverse Document Frequency.

The study carried out by (Haas and Derczynski, 2020) utilized a machine learning approach for automatic language identification among six Nordic languages: Danish, Swedish, Norwegian (Nynorsk), Norwegian (Bokmål), Faroese, and Icelandic. They developed the Nordic DSL (Distinguishing Similar Languages) dataset by compiling summaries from Wikipedia articles in each language. The study evaluated the performance of four classical models: K-Nearest Neighbors, Logistic Regression, Naive Bayes, and linear Support Vector Machine alongside two neural network architectures: Multilayer Perceptron and Convolutional Neural Network. The two best-performing models, FastText supervised and Convolutional Neural Network, showed decreased performance on off-domain data. However, utilizing character n-grams as features instead of words significantly improved the performance of the FastText Supervised classifier.

A state-of-the-art language identification system, surpassing previous models, is introduced by (Burchell et al., 2023a). This system achieves a macro-average F1 score of 0.93 across 201 languages. A FastText language identification model is developed that embeds character-level n-grams from the input text, which are then used as input to a multi-class linear classifier. This classifier was trained on 121 million lines of data covering 201 language classes sourced from various platforms such as news sites and Wikipedia.

3 Dataset

The dataset utilized in this study comprises five main components, each tailored to address specific challenges inherent in training and evaluating language identification models. The figure 1 illustrates the distribution of sentence counts per language across the five datasets utilized in this study.

3.1 Gold Training Dataset

We utilize the gold training/validation data sourced from Universal Dependencies treebanks (Nivre



Figure 1: Distribution of languages across datasets: Danish (da), Swedish (sv), Norwegian Nynorsk (nn), and Norwegian Bokmål (nb).

et al., 2017). While this dataset is clean, it lacks disambiguation. A sentence labeled as Nynorsk is likely to be in Nynorsk, but it could also be a valid Bokmål sentence.

3.2 Silver Training Dataset

Neural models have a tendency to utilize quick and possibly simplistic strategies whenever they are accessible. However, this behavior could introduce potential risks or drawbacks within the task at hand; for instance, if Norwegian datasets predominantly feature skiing-related texts and Swedish datasets ice hockey-related ones, the model might wrongly associate skiing-related sentences with Norwegian. In response to this challenge, we developed a synthetic training dataset termed "silver data," comprising machine-translated versions of all datasets in other languages investigated in this study. Integration of the silver training data with the existing gold training data yielded a balanced training corpus. This amalgamation ensures equitable distribution of sentence counts and domain representation across languages. However, it is important to note that this approach may introduce unintended artifacts. Artifacts in this context refer to unintended and potentially problematic elements introduced during the machine translation process. Artifacts

can distort the original meaning of the text, introduce inconsistencies, or create patterns that do not naturally exist in the original language data. This can be detrimental to training neural models as they might learn from these artificial patterns rather than genuine linguistic features, leading to poorer performance or incorrect associations. One example of an artifact introduced during machine translation is the mistranslation of idiomatic expressions. For instance, if the original text contains a phrase like "kick the bucket," which is an English idiom meaning "to die," the machine translation might translate it literally into another language, resulting in a nonsensical or incorrect translation. Figure 1 illustrates that the gold training and validation datasets are unbalanced towards Norwegian languages, having a higher number of samples in these languages compared to others. However, combining the silver and gold training datasets resulted in a new, balanced training dataset.

3.3 Manually Filtered Evaluation Dataset

The evaluation dataset is sourced from the test splits of the UD treebanks for each language, annotated with additional labels for sentences potentially belonging to multiple languages. For example, a sentence like "Hvordan går det?" could be both Bokmål and Danish, but not Nynorsk or Swedish.

4 Baselines

To establish a solid foundation for evaluating novel methodologies, we present a comprehensive suite of baseline methods in this study. These baselines serve as fundamental reference points against which we compare the performance of our proposed transfer-learning approaches.

The first baseline, OpenLID¹ (Burchell et al., 2023a), is a linear classifier built upon the FastText model. The model consists of 60.5 million parameters and achieves an inference time of 22.4 seconds over 206,448 lines of the test set (Burchell et al., 2023b).

The second and third baselines encompass the original FastText² (Joulin et al., 2016b,a) model and the updated NLLB language identifier³ (NLLB Team et al., 2022). Subsequently, we examined the langid⁴ (Lui and Baldwin, 2011, 2012; Heafield

¹<https://github.com/laurieburchell/open-lid-dataset>

²<https://fasttext.cc/docs/en/language-identification.html>

³<https://github.com/facebookresearch/fairseq/tree/nllblid-model>

⁴<https://github.com/saffsd/langid.py>

et al., 2015), lingua⁵, and langdetect⁶ models as additional baseline models.

5 Our proposed Transfer-learning approach

In this section, we present two pre-trained models for language identification that we fine-tuned specifically for Scandinavian language identification.

5.1 XLM-RoBERTa-base

XLM-RoBERTa⁷ (Conneau et al., 2019a), a multilingual iteration of RoBERTa, undergoes extensive pretraining on a massive dataset comprising 2.5 TB of meticulously curated CommonCrawl data spanning over 100 languages. Unlike conventional supervised methods, RoBERTa adopts a self-supervised approach, training on raw text data without human annotations, leveraging publicly available sources. Its pretraining process entails generating input-output pairs from text through an automatic process. Specifically, RoBERTa is pre-trained using the masked language modeling objective, where a fraction (15%) of words in a sentence are randomly masked, and the model predicts these masked words when given the entire masked sentence as input. This bidirectional approach allows RoBERTa to develop comprehensive representations of sentences in over 100 languages, enabling it to extract valuable features for diverse downstream tasks (Conneau et al., 2019b). The XLM-RoBERTa-base has been utilized, comprising 12 hidden layers, 12 attention heads, and a hidden size of 768. For both transfer-learning approaches, we applied the XLMRobertaTokenizerFast tokenizer. It is important to note that the Scandinavian languages examined in this study are among the 100 languages that the XLM-RoBERTa-base model is trained on.

5.2 XLM-RoBERTa-base-language-detection

This model is a refined version of the XLM-RoBERTa-base, specifically fine-tuned for language identification tasks⁸. It utilizes the transformer architecture of XLM-RoBERTa, supplemented with a classification head on top. The train-

⁵<https://pypi.org/project/lingua/>

⁶<https://github.com/Mimino666/langdetect>

⁷<https://huggingface.co/FacebookAI/xlm-roberta-base>

⁸<https://huggingface.co/papluca/xlm-roberta-base-language-detection>

Models	Loose Accuracy	Loose macro F1	Strict Accuracy	Strict macro F1	Runtime
NLLB	94.41 %	93.21 %	91.71 %	91.54 %	CPU: 0.39 (sec)
OpenLID	93.51 %	92.39 %	90.54 %	90.48 %	CPU: 0.51 (sec)
FastText	79.69 %	76.74 %	77.58 %	75.70 %	CPU: 0.10 (sec)
LangID	67.33 %	66.92 %	66.50 %	66.59 %	CPU: 10.06 (sec)
Lingua	91.71 %	91.17 %	88.72 %	89.26 %	CPU: 2.65 (sec)
langdetect	64.09 %	57.59 %	61.84 %	57.16 %	CPU: 25.60 (sec)

Table 1: Results of Scandinavian language identification using the six baseline approaches.

Models	Loose Accuracy	Loose macro F1	Strict Accuracy	Strict macro F1	Runtime	training dataset
XLM-RoBERTa-base	97.6 %	97.4 %	94.1 %	91.4 %	CPU: 93.41 (sec)	Only Gold training dataset
XLM-RoBERTa-base-language-detection	96.8 %	96.4 %	93.3 %	90.6 %	CPU: 120.25 (sec)	Only Gold training dataset
XLM-RoBERTa-base	96.1 %	96.4 %	93.0 %	90.4 %	CPU: 80.70 (sec)	Gold and silver training dataset
XLM-RoBERTa-base-language-detection	96.6 %	96.3 %	93.1 %	90.6 %	CPU: 92.50 (sec)	Gold and silver training dataset

Table 2: Results of Scandinavian language identification using transfer-learning approaches.

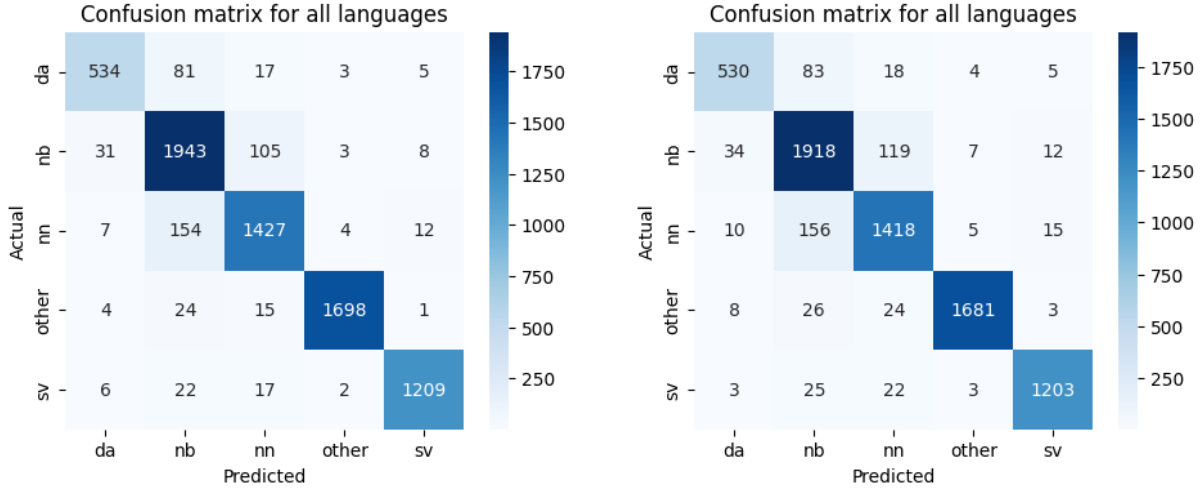


Figure 2: Confusion matrices of the fine-tuned version of XLM-RoBERTa-base and XLM-RoBERTa-base-language-detection based on the gold training dataset in five different categories: Danish (da), Swedish (sv), Norwegian Nynorsk (nn), Norwegian Bokmål (nb) and other.

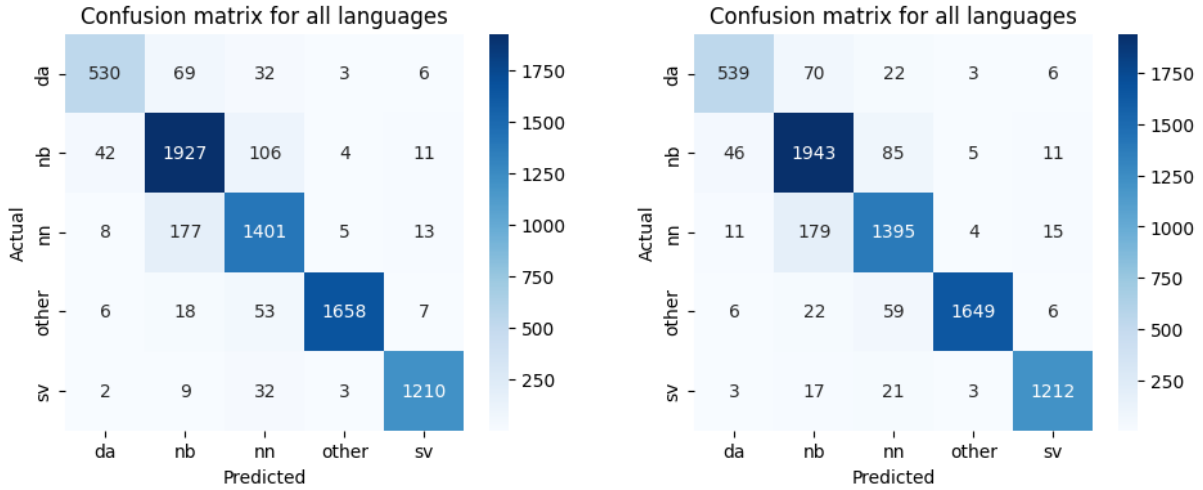


Figure 3: Confusion matrices of fine-tuned version of XLM-RoBERTa-base and XLM-RoBERTa-base-language-detection based on combination of the gold and silver training datasets in five different categories: Danish (da), Swedish (sv), Norwegian Nynorsk (nn), Norwegian Bokmål (nb) and other.

ing dataset comprises 70,000 samples, with both validation and test sets containing 10,000 samples each, spanning text sequences from 20 languages (Luca Papariello, 2024). We opted for the XLM-RoBERTa-base-language-detection model due to its specialization in language identification tasks, in contrast to the XLM-RoBERTa-base model, which was initially designed as a masked language model. However, it is worth noting that the Scandinavian languages investigated in this study were not included in the languages covered by the XLM-RoBERTa-base-language-detection model.

6 Results

In the upcoming section, we will present the results obtained from our baseline approaches and transfer learning models. These methods serve as benchmarks against which we evaluate the performance of our proposed transfer-learning approaches. In our analysis, we computed two variations of accuracy and average macro F1 score, termed as strict and loose.

6.1 Results based on baseline approaches

The outcomes derived from six baseline methodologies, encompassing the OpenLID, original FastText, updated NLLB, LangID, Lingua, and

langdetect language identifier models, are outlined through an evaluation conducted on the Manually Filtered Evaluation Dataset, as illustrated in Table 1. As depicted in Table 1, NLLB emerges as the top-performing model across all four performance metrics, although it ranks second in terms of runtime. FastText emerges as the most computationally efficient model, albeit occupying the fourth position in terms of performance. OpenLID and Lingua closely compete in terms of performance, securing the second and third positions, respectively, with OpenLID demonstrating fivefold faster processing. LangID and langdetect occupy the fifth and sixth positions, respectively, in terms of performance, with Lingua exhibiting almost twice the speed. Considering both performance and runtime aspects, NLLB emerges as the most favorable baseline method.

6.2 Results based on transfer-learning approach

We executed an assessment employing two distinct training configurations to examine the influence of the machine-translated Silver training dataset. Initially, we solely trained the model utilizing the gold training dataset and subsequently assessed its performance on the Manually Filtered Evaluation Dataset. In the second configuration, we amalgamated the silver and gold training datasets before evaluating the model on the Manually Filtered Evaluation Dataset, where the concatenation of the gold and silver training data result in a balanced dataset. This evaluation comprised two iterations: firstly, by fine-tuning the XLM-RoBERTa-base model, and secondly, by fine-tuning the XLM-RoBERTa-base-language-detection model. The latter model had been previously fine-tuned for identifying 20 languages; we further fine-tuned it specifically for Scandinavian language identification to leverage its existing knowledge of the aforementioned 20 languages.

The outcomes of the transfer learning strategies across all configurations are presented in Table 2. Upon comparing the results, it is evident that fine-tuning the XLM-RoBERTa-base model solely based on the gold training dataset surpasses other scenarios. This observation may suggest that the cross-lingual knowledge embedded in the XLM-RoBERTa-base-language-detection model did not contribute significantly to Scandinavian language identification. This could be attributed to the fact that this model is limited to supporting only 20

languages, none of which include Scandinavian languages. Consequently, it may lose some of the linguistic knowledge that the XLM-RoBERTa-base model possesses regarding these languages. Furthermore, it reflects the fact that fine-tuning a base model yielded better results than fine-tuning a model that has already been pre-trained on the same task but in other languages. This finding might be interesting because it contradicts the expectation that a pre-trained model, already familiar with similar tasks in other languages, would perform better than a base model. Additionally, the introduction of the machine-translated silver dataset did not improve model performance; instead, it reduced the runtime. In comparison between the baseline and transfer-learning approaches, the XLM-RoBERTa-base model achieves superior performance in terms of both loose and strict versions of accuracy and F1-score when compared to the baseline model. However, concerning runtime, the baseline model, NLLB operates approximately 240 times faster than the fine-tuned model. Figures 2 and 3 depict the confusion matrices of the four transfer learning approaches in identifying Scandinavian languages.

Discussion

This study compares various transfer learning approaches with existing baseline models for identifying Scandinavian languages. Two primary considerations for the current Scandinavian language identification task are model performance and computational efficiency, specifically runtime. Given that language identification serves as a fundamental preprocessing step for large corpora, minimizing computational resources is paramount.

Among the six baseline models evaluated on the held-out test dataset, the NLLB model emerged as the top performer. However, upon comparison with the best-performing fine-tuned model, XLM-RoBERTa-base trained solely on the gold training dataset, we observed that while NLLB excelled in runtime efficiency, the fine-tuned model improved all four metrics: strict and loose accuracy, as well as F1-score.

This study has several limitations. The models were trained and evaluated on datasets that may not fully capture the linguistic diversity and nuances of Scandinavian languages. Additionally, the computational resources required for fine-tuning large models like XLM-Roberta can be prohibitive, limiting accessibility for researchers with less com-

putational power.

Future research should explore several areas to enhance Scandinavian language identification further. One potential direction is to investigate the use of Scandinavian models, such as NorBERT (Samuel et al., 2023), ScandicBERT (Snæbjarnarson et al., 2023). Additionally, incorporating more diverse datasets, including social media text and informal language, could improve model robustness.

Conclusion

In conclusion, while the fine-tuned XLM-RoBERTa-base model showed improvements in performance metrics, these gains must be balanced against computational runtime.

References

- Ralf D Brown. 2013. Selecting and weighting n-grams to identify 1100 languages. In *Text, Speech, and Dialogue: 16th International Conference, TSD 2013, Pilsen, Czech Republic, September 1-5, 2013. Proceedings 16*, pages 475–483. Springer.
- Laurie Burchell, Alexandra Birch, Nikolay Bogoychev, and Kenneth Heafield. 2023a. [An open dataset and model for language identification](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 865–879, Toronto, Canada. Association for Computational Linguistics.
- Laurie Burchell, Alexandra Birch, Nikolay Bogoychev, and Kenneth Heafield. 2023b. An open dataset and model for language identification. *arXiv preprint arXiv:2305.13820*.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019a. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. [Unsupervised cross-lingual representation learning at scale](#). *CoRR*, abs/1911.02116.
- René Haas and Leon Derczynski. 2020. Discriminating between similar nordic languages. *arXiv preprint arXiv:2012.06431*.
- Kenneth Heafield, Rohan Kshirsagar, and Santiago Barona. 2015. [Language identification and modeling in specialized hardware](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 384–389, Beijing, China. Association for Computational Linguistics.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Herve Jégou, and Tomas Mikolov. 2016a. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016b. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.
- Luca Papariello. 2024. [xlm-roberta-base-language-detection \(revision 9865598\)](#).
- Marco Lui and Timothy Baldwin. 2011. Cross-domain feature selection for language identification. In *Proceedings of 5th international joint conference on natural language processing*, pages 553–561.
- Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30.
- Joakim Nivre, Daniel Zeman, Filip Ginter, and Francis Tyers. 2017. Universal dependencies. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*.
- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Hefernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia-Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation.
- David Samuel, Andrey Kutuzov, Samia Touileb, Erik Velldal, Lilja Øvrelid, Egil Rønningstad, Elina Sigdel, and Anna Palatkina. 2023. Norbench—a benchmark for norwegian language models. *arXiv preprint arXiv:2305.03880*.
- Vésteinn Snæbjarnarson, Annika Simonsen, Goran Glavaš, and Ivan Vulić. 2023. Transfer to a low-resource language via close relatives: The case study on faroese. *arXiv preprint arXiv:2304.08823*.
- Marcos Zampieri, Liling Tan, Nikola Ljubešić, and Jörg Tiedemann. 2014. A report on the dsl shared task 2014. In *Proceedings of the first workshop on applying NLP tools to similar languages, varieties and dialects*, pages 58–67.

Marcos Zampieri, Liling Tan, Nikola Ljubešić, Jörg Tiedemann, and Preslav Nakov. 2015. Overview of the dsl shared task 2015. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, pages 1–9.

Scandinavian Language Identification

Jonas Sebulon Frydenberg
jonassf@ifi.uio.no

Abstract

Language identification is by many considered a solved task today. Performance on minor and closely related languages is, however, still lacking. An additional important factor of a good language identification model is its speed, due to the application of language identification in the preprocessing of large-scale corpora. The goal of this paper is to improve language identification on the closely related Scandinavian languages. We compare existing approaches and present finetuned BERT-like models (NorBERT, ScandiBERT, XLM-R) that reaches great performance at the cost of longer inference time, as well as a fast hybrid approach that combines the speed of a FastText model with the performance of a deep learning model.

1 Introduction

Language identification (LID) is an important part of the preprocessing step for large-scale text corpora, like web-crawls. An accurate language identification model is thus necessary to ensure that the data sets are clean, in the sense that they do not contain data from other languages. During the gathering and processing of web-crawled corpora, several terabytes of data have to be processed, the inference time of such a LID model needs to be as fast as possible.

Language identification for major languages is by many considered a solved task, but there leaves a lot to be desired when dealing with minor languages, especially closely related ones like the Scandinavian languages (Norwegian, Swedish and Danish), which is the topic of this paper.

One can distinguish between language identification performed at the document-level and on the sentence-level. Document-level identification is a much easier task than sentence-level identification due to more data and thus a higher likelihood of finding distinguishing elements. On the sentence level, there are fewer words and less information

that can be used to tell languages apart, let alone the fact that short sentences have a higher likelihood of being passable in several languages. The topic of this paper will be sentence level classification. Important applications of such a task include language identification for parallel corpora intended for machine translation and language identification for short sentences in a search engine like Google.

The rest of the paper is structured as follows. Section 2 provides some details on the different Scandinavian languages. In Section 3, the dataset splits and their sources are described. Related work and existing methods are described in Section 4. In Section 5, we explore different approaches to language identification and their performance. A baseline evaluation is established in 5.1. In 5.2, we explore a dictionary-based approach, while in Section 5.3, we explore the usage of deep learning models as well as a hybrid architecture in 5.4. A discussion of the the different models is provided in section 6. We briefly point out directions for future work in Section 7, before concluding in Section 8.

2 Background

The languages considered for this task will be Norwegian Nynorsk, Norwegian Bokmål, Danish, Swedish and "other" (which means none of the Scandinavian languages). These four languages are all part of the North-Germanic branch of the Germanic languages. Traditionally, Norwegian along with Icelandic and Faroese have been classified as West-Nordic while Danish and Swedish are East-Nordic.

Norwegian Bokmål however is a continuation of the written Danish that was used in Norway from the 1600s to early 1900s. This places it closer to Danish, which is also reflected in formal Bokmål being very close to Danish. Norwegian Nynorsk has traditionally been kept quite distinguished from Bokmål, but the official spelling reform of 2012 has

made it possible to write Bokmål-close Nynorsk. While Norwegian (both varieties) and Danish use the same alphabet, Swedish has its own unique letters "ä" and "ö" and an orthographic convention that makes it stand out from the other three. Automatic language identification on Swedish is thus easier than on Bokmål, Nynorsk, Danish.

Some issues are sentences which are similar in the different languages as well as sentences which are grammatically sound in several languages, but with different meaning. Some examples of this from the test set (see 3.1) are:

- Hun skulle betale for bogen.
- Har du skrevet en bog?.

The first sentence is a grammatically valid and semantically meaningful sentence in Nynorsk, Bokmål and Danish. The difference is that the Danish word *bogen* means 'the book', while the Bokmål and Nynorsk word *bogen* means 'chuck' (meat from the shoulder of an animal). The second is *grammatically valid* in Bokmål in the sense that a noun is the object of a verb, but semantically it is nonsensical ("Have you written an *animal shoulder*"). A word-wise lexical lookup approach even if combined with POS tagging, would not be able to classify the second sentence as Danish only (it could be remedied with n-gram statistics but such an approach will always be limited by training data and will thus not scale well to the unseen). A transformers model could however model the relationship between *skrevet* and *bogen* and recognise that *bogen* in this case is not likely to mean 'animal shoulder'. A third scenario is the following sentence which is also taken from the test set (in which it is labelled as Nynorsk):

- Vann folket med plyndring.

This is likely the title of a news article in which the subject is omitted. This makes it seem like the verb is in the imperative mood. The first interpretation yields the Nynorsk sentence: "Won the people by plunder.", where *vann* is the past tense of *å vinne* 'to win', while the latter interpretation yields the Bokmål sentence: "Water the people with plunder!", where *vann* is the imperative of the verb *å vanne* 'to water'. It is not trivial to conclude whether it should be considered correct to classify the above sentence as Bokmål and Nynorsk or only Nynorsk.

3 The Data

In this section, the different datasets and their sources will be described. In Section 5.3, we will experiment with finetuning deep learning models on combinations of the following datasets.

3.1 Universal Dependencies Treebank

The provided datasets come from the Universal Dependencies (UD) Treebank (Nivre et al., 2017), see table 1 for complete statistics. The gold data set, which is taken directly from the UD treebank is skewed with about 30 000 sentences in Bokmål and Nynorsk compared to about 8500 sentences in Danish and Swedish. An additional "silver" training set, which is also based on the UD treebank data, tries to remedy the skewed data distribution with machine translated content.

The provided test set, which consists of the UD treebank test splits for each of the included languages, has been manually annotated with additional labels when the sentences could belong to multiple languages. There are some issues with the provided test set, however. Upon manual inspection, we changed 55 sentences and removed two¹. For the most part, sentences which were valid in more than one language, but had only one language tag, had to be updated with additional tags (especially Bokmål sentences that were also valid in Danish or Nynorsk). Two sentences contained a quote in Nynorsk, but they were tagged as Bokmål.

3.2 Other

In addition to the UD data, we gathered data from the Tatoeba corpus² (Tiedemann, 2020) and the Open LID dataset³ (Burchell et al., 2023), as well as some extra samples of "other" languages from the Bokmål antgiold test set used by FastSpell⁴. It is important that the data comes from reliable sources in which the data will actually be in the language it claims to be. Web crawled data is therefore excluded from consideration, since they are noisy and full of erroneous language identification even if they have been cleaned.

¹Full list of changes is available here: https://github.com/uio-no/jonassf/in5550_exam/blob/main/data/test_fixed_log.txt

²https://huggingface.co/datasets/Helsinki-NLP/tatoeba_mt

³<https://github.com/laurieburchell/open-lid-dataset?tab=readme-ov-file>

⁴<https://github.com/mbanon/benchmarks/tree/main/langid/testsets/antigold>

Language	Gold train	Tatoeba	openLID	Other	Gold valid	Silver train	Evaluation set
Bokmål	15 435	17 636	35 000	—	2 392	22 860	1 938
Danish	4 376	58 400	1 538	—	563	34 173	564
Nynorsk	14 029	1 640	35 000	—	1 859	24 382	1 511
Swedish	4 278	53 053	1 541	—	503	34 252	1 219
Other	8 805	—	—	500	1 165	—	1 742

Table 1: Number of sentences per language per corpus

Here follows descriptions of the additional data sets.

Tatoeba Tatoeba is a collection of translated sentences based on voluntary contributions. We include all the data from Tatoeba for Bokmål, Nynorsk, Danish and Swedish. The Tatoeba dataset contains short, informal and colloquial sentences. This is ideal as an additional data set for sentence-level classification as most other data sources will contain formal language. The language distribution is on the other hand skewed, with only 1 640 sentences in Nynorsk, and over 50 000 sentences for both Danish and Swedish.

openLID The full openLID dataset is 21GB of data in 201 languages. We extract the Nynorsk, Bokmål, Danish and Swedish parts of it and remove the sentences with the source tag MT560, (except for the Bokmål part for which MT560 is the only data source) as we want to ensure high quality sentences without localization files. The MT560 dataset (Gowda et al., 2021) contains a sizeable amount of data from the Bible. Archaic Bokmål bible verses are written in stilted language which closely resembles Danish, an example is the usage of the archaic second person plural pronouns *I/eder/eders* which in my opinion should not be a part of the Bokmål data. Obvious biblical sentences are thus filtered out by a simple regular expression⁵ that matches some common biblical words as well as the "[chapter]: [verse]" formula for citations from the Bible.

By randomly picking out sentences, 35 000 sentences from the open LID dataset for Bokmål and Nynorsk, and about 1500 for Danish and Swedish are gathered. The aim of this skewed distribution is to balance out the language distribution of the Tatoeba data.

other The antigold Bokmål part of the benchmark data used by FastSpell contains sentences

⁵(jer|eder|eders|kristen|jehova|disipler|[0-9]+:[0-9]+)

from languages other than Bokmål. The sentences are sorted by language, and Danish and Nynorsk sentences were manually removed. This dataset is used to add further examples of "other" languages.

4 Related work

The best performing fast language identification models are simple linear classifiers trained on top of a FastText (Bojanowski et al., 2016) embedding model. These include the openLID model (Burchell et al., 2023), the original FastText model⁶ and the updated and extended No Language Left Behind (NLLB) language identifier ((Team et al., 2022); (Joulin et al., 2017)). FastText learns word embeddings by extending the continuous skip-gram model (Mikolov et al., 2013) with subword information by representing each word as the sum of the vectors of the n-grams that constitute it. CLD3⁷ is made by Google and works similarly to FastText.

Other approaches include dictionary based word lookup ,in which lexica enriched with inflected forms of all words are used to identify the language by outputting the language with the most dictionary hits. This approach is heavily limited by the quality of the underlying dictionary. It also falls short when dealing with named entities (of which an exhaustive list will never be included in the underlying dictionary) and homographs with different meanings in different language for which grammatical information is ignored when using bag of words majority voting. More details about this are discussed in 5.2.

Fastspell (Bañón et al., 2024) is the LID model used in Bicleaner, which has been used to clean the enormous web-crawled corpus ParaCrawl (Esplà et al., 2019). It is a hybrid approach that combines a FastText model with dictionary-based spell checking using Hunspell⁸. It works by first defining sets

⁶<https://fasttext.cc/docs/en/language-identification.html>

⁷<https://github.com/google/cld3>

⁸<https://hunspell.github.io/>

of closely related languages. When the FastText model predicts a language that is a part of one of these sets, spell checking is performed to determine exactly which of the closely related languages it is. The FastSpell tool uses the original 176-language FastText model which, as seen in table 2, performs rather poorly compared to newer models. The spell checking also faces the issues discussed in the paragraph above. Since FastSpell is aimed at parallel corpora filtering, it requires a target language when the model is instantiated. We evaluated the model using *da*, *nn*, *nb*, *no*, and *sv* and report the best result obtained when using *nb* as the target. This means that when Danish, Bokmål and Nynorsk would get an equal score using Hunspell, *nb* will be output.

5 Methodology

5.1 Baseline

The models will be evaluated in two ways: strict and loose. The test set contains data with potentially multiple labels, due to the sentences being valid in more than one language. The loose metric considers a prediction correct if the prediction is a subset of the gold labels. The strict metric on the other hand, only considers a prediction correct if it exactly matches the gold labels. Most of the available language identification tools only output one language, although confidence scores and top-n predictions are available for some of them. In this paper, the loose metric will be used to evaluate single-label prediction and the strict metric to evaluate multi-label prediction.

In table 2, F1 and MCC (Matthews correlation coefficient) scores for a selection of available language identification methods are reported. The best score for each metric is highlighted in boldface. There are two methods that stand out: *NLLB* and *FastSpell*, with *NLLB* beating *FastSpell* in the overall macro average, as well as on Bokmål and Nynorsk. *FastSpell* somewhat surprisingly delivers strong results on Danish, Swedish and other classification. Both of these utilize FastText embeddings and a linear classifier head, but *FastSpell* uses dictionary based spell checking as well. *NLLB* is extremely fast, averaging 0.10ms runtime per sentence in the test set. *FastSpell* is however slightly slower with an average of 0.43 ms/sentence. As is evident from the table, they clearly outperform the original 176 language FastText model, which only reaches a macro F1 score of 77.04. On the

other hand, the original FastText model is even faster than *NLLB* at 0.02 ms/sentence. A notable mention is *Lingua*, which uses rule-based and statistical methods. It reaches a macro F1 score of 91.53, making it the fourth best baseline method. Even though it runs quite fast, it is significantly slower than *NLLB* and *FastText* at 0.71 ms / sentence.

5.2 Dictionary based

In this section we will explore dictionary based language identification and combine it with a POS tagger.

We gather full morphological dictionaries (wordbanks) of Bokmål⁹, Nynorsk¹⁰ and Danish¹¹. The data is then read and filtered, notably words containing numbers, proper nouns and symbols are removed. In addition, plural and definite past participles of intransitive verbs in Bokmål are removed. This is due to their similarity to Danish past tense verbs ending in *-ede*. Since past participles of intransitive verbs inflected for plural or definite are very seldom, if at all, used, they can safely be removed. Sentences are then classified by matching each word in the sentence with each target language’s wordbank, the language with the most matched tags is then returned. If there is a tie, several languages are returned.

This approach is fast (0.18 ms/sentence) and works fairly well for long text segments or documents, but it often fails due to words of different parts of speech being equal to different words in another language. For instance the distinctive Danish second person plural pronoun *i ’you’* is equal to the Norwegian (and Danish) preposition *i ’in’*. This approach can not distinguish between words in such cases. It also fails at novel compound words which are not in the dictionary, as well as named entities.

To try to remedy this, we train a POS-tagger using a bidirectional LSTM-CRF on the Norwegian Bokmål, Norwegian Nynorsk, Danish and Swedish UD treebanks (training and development data combined) for 90 epochs using the Flair NLP toolkit¹² (Akbik et al., 2019). The tagger gets a micro F1 score of 0.97 on the test-splits of the Bokmål, Nynorsk, Danish and Swedish UD treebanks.

⁹<https://www.nb.no/sprakbanken/ressurskatalog/oai-nb-no-sbr-5/>

¹⁰<https://www.nb.no/sprakbanken/en/resource-catalogue/oai-nb-no-sbr-41/>

¹¹<https://korpus.dsl.dk/resources/details/ddo-fullforms.html>

¹²<https://github.com/flairNLP/flair>

Method	Metric	nb	nn	da	sv	other	Macro average
Runtime (ms / sent)							
Random	F1	24.11	23.50	12.28	22.14	24.35	21.27
<0.01 ms	MCC	28.76	29.65	22.03	29.31	29.67	27.88
CLD3	F1	69.10	0.00	79.44	92.82	85.69	65.41
0.15 ms	MCC	60.75	1.64	79.49	91.60	82.06	63.11
NLLB	F1	94.40	93.58	89.98	97.03	93.40	93.68
0.10 ms	MCC	92.39	92.19	89.54	96.45	91.61	92.44
FastText	F1	73.13	59.08	66.96	92.80	93.20	77.04
0.02 ms	MCC	67.44	59.96	68.02	91.54	91.23	75.64
OpenLID	F1	93.50	92.77	87.73	96.12	93.17	92.66
0.10 ms	MCC	91.25	91.20	87.35	95.37	91.39	91.31
LangID	F1	28.52	78.26	70.64	94.40	62.40	66.85
1.87 ms	MCC	34.93	76.01	71.62	93.40	53.56	65.90
Lingua	F1	90.31	90.00	89.29	95.85	92.21	91.53
0.71 ms	MCC	87.30	87.98	88.85	95.06	90.24	89.88
Langdetect	F1	64.76	0.00	66.75	89.05	68.92	57.90
8.35 ms	MCC	54.54	1.68	67.89	87.29	66.83	55.65
FastSpell	F1	92.72	90.45	90.12	98.01	94.02	93.06
0.43 ms	MCC	90.03	88.78	89.77	97.61	92.28	91.69

Table 2: Baseline results with runtime per sentence, F1 and MCC scores in percentages for each language, and the macro average to the right

In practice however, it often misclassifies subject complements: in the sentence *Vi er arabere* ‘We are arabs’, the tagger classifies *arabere* as an adjective. It also has issues with very short and/or incomplete sentences (like news titles). Overall, it works decently.

The wordbanks are then read, and words are stored in a dictionary with universal POS tags as the keys. Some ambiguous words are removed, like the rare Norwegian verb *blive* ‘to drown’ which is equal in form and conjugation to the very common Danish copulative and auxiliary verb *blive* ‘to become, to be (forms the passive voice)’ — as they are both verbs, this model can’t distinguish them. When classifying a sentence, the POS tagger assigns a POS tag to each word, which is then matched to the set of words in that POS tag for each language. Due to the unreliable POS tagger, if none of the wordbanks match a word of a particular POS, the word is matched against the full wordbank (without considering parts of speech). We then apply a simple scoring system: if the POS-tagged word is only found in one language, that word is worth 1.5 points, if it is found in >1 language, it is worth 1.0 point, finally, words that have to be matched against the full wordbank are worth 0.5 points.

Even though performance is slightly improved, it comes at the cost of significantly longer inference time (23.08ms/sent). It also fails to distinguish words of the same POS but with totally different semantical meanings, see 2 for a concrete example.

The wordbank classifier only has data for Bokmål, Nynorsk and Danish. As a consequence, they can not reliably detect that a sentence is not in one of these languages — there will likely be words in the dictionaries that match words in foreign languages. Similarly to FastSpell’s approach, we combine the wordbank classifier with a FastText model which is first used to make a language prediction, if this language is Bokmål, Nynorsk or Danish, the wordbank classifier is used to determine which of these it is, else the FastText model prediction is returned.

One advantage with this approach is that it is inherently multi-label. This can also be seen in table 3, where both the models have a higher score for the more difficult strict evaluation.

5.3 Deep learning

Since the target of this task is Scandinavian language identification, it is necessary to deploy models pretrained on these languages. Three different pre-trained models will thus be considered:

Eval	Method	F1	MCC	Overlap F1
Loose	Wb	87.08	84.17	—
	Wb+POS	89.30	87.11	—
Strict	Wb	89.94	87.58	93.56
	Wb+POS	90.66	88.51	93.90

Table 3: Performance of the dictionary based Wordbank (Wb) and Wordbank with POS tagging (Wb+POS) with macro F1 and MCC scores using both loose and strict evaluation

- *NorBERT*₃ (Samuel et al., 2023a): A family of BERT-like models trained on Norwegian *only* data using the optimized training methods as described in (Samuel et al., 2023b). *NorBERT*₃ comes in four sizes:
 - *xs* (L = 12, H = 192, A = 3, 15M params)
 - *small* (L = 12, H = 384, A = 6, 40M params)
 - *base* (L = 12, H = 768, A = 12, 123M params)
 - *large* (L = 24, H = 1024, A = 16, 353M params)

All four of them will be evaluated in this paper. It is worth noting that these models use a Word-Piece subword tokenizer (Wu et al., 2016) with a vocabulary size of 50 000, which is trained only on Norwegian data.

- *ScandiBERT* (Snæbjarnarson et al., 2023): A Scandinavian BERT model (L = 12, H = 768, A = 12, 125M params) trained on Danish, Faroese, Swedish, Icelandic and Norwegian data. The majority of the data is however Norwegian and Icelandic — with Norwegian accounting for roughly 63% and Icelandic accounting for approximately 24%.¹³
- *XLM-RoBERTa* (Conneau et al., 2020): A multilingual roberta-model trained on 2.5TB of filtered common crawl data containing 100 languages (including Norwegian, Danish and Swedish). We consider the *XLM-RoBERTa*_{base} model (L = 12, H = 768, A = 12, 270M params).

5.3.1 Finetuning

The pretrained models described in section 5.3 were finetuned on various combinations of the training data sets described in section 3.2. The models

were trained for 5 epochs, with a batch size of 32, a learning rate of $1e^{-4}$ using 100 warmup steps and a linear scheduler together with the AdamW optimizer.

The best performing models were trained on a combination of the Tatoeba and gold datasets described in 3.2. The performance of the models trained on all the training data (Tatoeba, open LID, gold, other) is shown in table 6. These models reached competitive performance, but they were slightly edged out by the models trained only on Tatoeba and the gold train set. One would believe that more data equals better performance, and also that similar distributions of the different labels would be beneficial. The best models presented in this paper proves otherwise for this particular task. Models trained on the silver dataset, as well as combinations of the silver dataset with Tatoeba and open LID were worse than models trained on the gold set by a substantial margin (around 10 percentage points difference in F1). Their ability to discriminate Bokmål and Nynorsk worsened. It seems that the unbalanced data distribution is beneficial for the model, at least for the single label prediction task. The impact of finetuning on the silver dataset on the multi label prediction task has not been fully explored.

The results shown in table 4 show that the *xs* and *small* versions of *NorBERT* have very competitive and even better performance than the larger models. This is rather unexpected, it is likely that more and/or different training data as well as further hyperparameter tuning would improve the larger models.

A way to improve the performance of a smaller and faster model is knowledge distillation (Sanh et al., 2020) by using a slow "teacher" model to guide the faster "student" model. Due to the great performance of directly finetuning *NorBERT*_{xs} however, such a training approach was not necessary.

Inference speed is a crucial aspect of a language identification tool. In the evaluation script, the test set is evaluated sentence for sentence, i.e. without batching. We also choose to perform the evaluation on CPUs only — all the running times reported in table 4 are measured on the HPC-Cluster fox¹⁴ using 24 CPUs and 48 GB of RAM. The FastText models are intended to be used on CPUs, and since

¹³Training procedure and model available here: <https://huggingface.co/vesteinn/ScandiBERT>

¹⁴<https://www.uio.no/english/services/it/research/hpc/fox/>

Method	Metric	nb	nn	da	sv	other	Macro average
Runtime (ms / sent)							
NorBERT3_{large}	F1	96.84	96.75	94.92	98.09	97.76	96.87
	130.57 ms	MCC	95.69	95.93	94.57	97.71	96.19
NorBERT3_{base}	F1	97.95	97.97	94.93	99.02	98.44	97.66
	44.38 ms	MCC	97.17	97.45	94.59	98.82	97.19
NorBERT3_{small}	F1	98.34	98.40	95.69	99.02	98.49	97.99
	23.91 ms	MCC	97.71	97.97	95.38	98.82	97.58
NorBERT3_{xs}	F1	98.25	98.36	95.03	98.41	98.02	97.61
	17.71 ms	MCC	97.58	97.92	94.70	98.09	97.14
RoBERTa	F1	96.81	96.46	94.59	98.46	97.96	96.85
	31.16 ms	MCC	95.63	95.59	94.23	98.14	96.18
ScandiBERT	F1	97.58	97.28	95.68	98.73	97.86	97.43
	28.32 ms	MCC	96.68	96.58	95.38	98.47	96.86

Table 4: Results of the finetuned models with runtime per sentence, F1 and MCC scores in percentages for each language, and the macro average to the right.

Model	T	F1	MCC	speed
NorBERT3_{xs}	0.25	93.99	92.79	0.67ms
NorBERT3_{xs}	0.5	95.06	94.06	1.46ms
NorBERT3_{xs}	0.80	96.93	96.30	2.95ms
NorBERT3_{xs}	0.95	97.60	97.11	4.46ms
NorBERT3_s	0.25	94.04	92.85	0.99ms
NorBERT3_s	0.5	95.18	94.20	2.04ms
NorBERT3_s	0.80	97.13	96.52	4.22ms
NorBERT3_s	0.95	97.95	97.52	6.22ms
NorBERT3_b	0.25	94.03	92.84	1.25ms
NorBERT3_b	0.5	95.09	94.10	3.98ms
NorBERT3_b	0.80	96.95	96.31	7.53ms
NorBERT3_b	0.95	97.61	97.11	11.46ms

Table 5: Macro average F1 and MCC as well as inference time (ms/sentence) for hybrid NorBERT models using different thresholds (T).

they are the models to beat, inference speed should be measured against them in a similar fashion. For large-scale text corpora filtering, it would be beneficial to do batched prediction on GPUs. This would significantly improve the inference speed of the slow deep learning models. GPUs are however not always available. In section 5.4, a hybrid approach with much improved inference speed on CPUs without batching will be described.

5.4 Hybrid

Even though the transformer models beat the other approaches by quite a margin in terms of pure performance, it comes at the cost of a significantly longer inference time. To remedy this, we will

combine the best performing FastText model — NLLB — with the finetuned NorBERT3_{small} and NorBERT3_{xs}, which are compact models with great performance, into a hybrid model. It works by first predicting the language of a sentence using the fast NLLB model, and only falling back to the NorBERT3-model in when it is uncertain, i.e. the confidence score is low. Testing has shown that a threshold of 0.80, which means that sentences with a FastText confidence score below 0.80 will be predicted by the NorBERT3-model, is a decent compromise between speed and performance. This threshold can naturally be altered by use case. See table 5 and table 7 in the appendix for a comprehensive threshold, score and speed overview.

Using a threshold of 0.25, the NorBERT3_{xs} hybrid LID model uses just 0.67 ms/sentence, but only at a marginal performance increase from the pure NLLB model: F1 from 93.68 to 93.99. There is always a trade-off between speed and performance, the NorBERT3_{xs} hybrid model with a threshold of 0.80 seems to be the best performer for the speed. It reaches an F1 score of 96.93 and MCC 96.30 at an inference speed of 2.95 ms. The NorBERT3_{small} hybrid model with a threshold of 0.95 hardly sacrifices any performance compared to the NorBERT only model (F1 97.95 vs 97.99) being four times faster (6.22ms/s vs 23.91ms/s).

6 Discussion

The NLLB FastText model is exceptionally fast at 0.10 ms/sentence and delivers great performance with a macro F1 score of 93.68 on single label pre-

diction on the test set. FastSpell is not far behind with a macro F1 score of 93.06. These scores are, however, slightly inflated by their performance on predicting Swedish and "other". The most challenging part is distinguishing Bokmål, Nynorsk and Danish. Per language metrics reveal that NLLB only reaches F1 89.98 on Danish — more impressive is F1 94.40 on Bokmål and 93.68 on Nynorsk — while FastSpell reaches F1 92.72, 90.45 and 90.12 on Bokmål, Nynorsk and Danish respectively.

The step up from FastText embeddings is deep learning models. We have evaluated the multilingual models NorBERT3, ScandiBERT and XML-RoBERTa. Even though ScandiBERT and NorBERT3 score similarly, the NorBERT3 in the small and xs variants deliver slightly better performance in a much smaller and faster package. The best performing model, NorBERT3_{small}, reaches F1 98.34 on Bokmål and 98.40 on Nynorsk. This shows a significant improvement over the FastText models mentioned above. It's weakest language is Danish, with an F1 score of 95.69. This could be due to the model preferring to predict Bokmål in cases where it is tied with Danish and in sentences that are hard to classify in which only a single letter distinguishes the two languages — some examples are the *-t*-suffix that Danish uses in adjectives ending in *-ig* inflected for neuter, which Norwegian does not, and the letter *j* in for instance Danish *køkken* versus Norwegian *kjøkken* 'kitchen'.

A simple dictionary-based approach is not that useful compared to the other available models, even when paired with a POS tagger. Efforts could be put into an improved POS tagger and further word-bank filtering to identify potentially distinguishing words that are not exploited due to the presence of rare and/or obscure words that are similar to common words. This is a laborious task and is probably not worth the effort compared to other approaches.

7 Future work

We did not fully explore the effects of multi-label classification. The majority of the test data contains single-labeled sentences. When evaluating the models on multi-label prediction, we often experienced deteriorated performance — the single-label prediction models got better results on the strict metric than models with multi-label prediction, which should not be the case. Experimentation with different activation functions, like softmax and sigmoid, and different thresholds for when

to output multiple labels yielded inconclusive results. More work needs to be done to fully explore the impact of more training data and the effect on the model's multi-label prediction abilities.

8 Conclusion

In this paper, we have evaluated existing methods for automatic language identification and compared them to dictionary based methods and well-performing finetuned deep learning models. The best existing methods are the speedy FastText models. In this paper, we have found a finetuned NorBERT3_{small} model to be the top performer. To leverage the high performance of the transformers models and the outstanding speed of the FastText models, we propose a hybrid architecture. This hybrid approach reaches competitive speed and a great performance, the FastText/NorBERT3-threshold can be adapted to balance speed vs performance in differently demanding applications.

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.
- Marta Bañón, Jaume Zaragoza-Bernabeu, Gema Ramírez-Sánchez, and Sergio Ortiz-Rojas. 2024. FastSpell: the LangId Magic Spell. `_eprint: 2404.08345`.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomáš Mikolov. 2016. [Enriching Word Vectors with Subword Information](#). *CoRR*, abs/1607.04606. ArXiv: 1607.04606.
- Laurie Burchell, Alexandra Birch, Nikolay Bogoychev, and Kenneth Heafield. 2023. [An Open Dataset and Model for Language Identification](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 865–879, Toronto, Canada. Association for Computational Linguistics.
- 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.

- Miquel Esplà, Mikel Forcada, Gema Ramírez-Sánchez, and Hieu Hoang. 2019. [ParaCrawl: Web-scale parallel corpora for the languages of the EU](#). In *Proceedings of Machine Translation Summit XVII: Translator, Project and User Tracks*, pages 118–119, Dublin, Ireland. European Association for Machine Translation.
- Thamme Gowda, Zhao Zhang, Chris A Mattmann, and Jonathan May. 2021. [Many-to-english machine translation tools, data, and pretrained models](#). *Preprint*, arXiv:2104.00290.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality.
- Joakim Nivre, Daniel Zeman, Filip Ginter, and Francis Tyers. 2017. [Universal Dependencies](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, Valencia, Spain. Association for Computational Linguistics.
- David Samuel, Andrey Kutuzov, Samia Touileb, Erik Velldal, Lilja Øvrelid, Egil Rønningstad, Elina Sigdel, and Anna Palatkina. 2023a. [NorBench – A Benchmark for Norwegian Language Models](#). In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 618–633, Tórshavn, Faroe Islands. University of Tartu Library.
- David Samuel, Andrey Kutuzov, Lilja Øvrelid, and Erik Velldal. 2023b. Trained on 100 million words and still in shape: BERT meets British National Corpus. *_eprint*: 2303.09859.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *_eprint*: 1910.01108.
- Vésteinn Snæbjarnarson, Annika Simonsen, Goran Glavaš, and Ivan Vulić. 2023. Transfer to a Low-Resource Language via Close Relatives: The Case Study on Faroese. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, Tórshavn, Faroe Islands. Linköping University Electronic Press, Sweden.
- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Hefernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No Language Left Behind: Scaling Human-Centered Machine Translation. *_eprint*: 2207.04672.
- Jörg Tiedemann. 2020. [The Tatoeba Translation Challenge – Realistic Data Sets for Low Resource and Multilingual MT](#). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online. Association for Computational Linguistics.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *_eprint*: 1609.08144.

A Appendix

Method Runtime (ms / sent)	Metric	nb	nn	da	sv	other	Macro average
NorBERT3_{small} all 22.87 ms	F1	97.79	98.20	94.90	98.61	98.10	97.52
	MCC	96.96	97.73	94.57	98.33	97.52	97.02
NorBERT3_{xs} all 15.64 ms	F1	97.69	97.41	95.15	98.65	98.64	97.51
	MCC	96.84	96.72	94.83	98.37	98.21	96.99
NorBERT3_{xs} (sm) all 15.71 ms	F1	98.10	98.11	96.21	98.57	98.61	97.92
	MCC	97.38	97.60	95.94	98.28	98.17	97.47
ScandiBERT all 26.04 ms	F1	97.17	97.13	94.88	98.73	97.21	97.02
	MCC	96.10	96.40	94.55	98.47	96.36	96.38
ScandiBERT64 all 27.51 ms	F1	97.31	97.48	96.43	98.90	97.72	97.57
	MCC	96.29	96.83	96.17	98.67	97.03	97.00
XLm-RoBERTa all 30.99 ms	F1	87.44	70.28	80.86	98.17	96.25	86.60
	MCC	83.22	69.01	80.64	97.80	95.17	85.17

Table 6: Performance of models finetuned on all datasets (Gold, Tatoeba, openLID, other). Note that the NorBERT3_{xs}(sm)all was trained on a smaller openLID dataset with only 33 079 sentences. The ScandiBERT64 model was trained with a batch size of 64

Model	Threshold	Macro F1	Macro MCC	speed
NorBERT3_{xs}	0.25	93.99	92.79	0.67ms
NorBERT3_{xs}	0.5	95.06	94.06	1.46ms
NorBERT3_{xs}	0.75	96.74	96.05	2.98ms
NorBERT3_{xs}	0.80	96.93	96.30	2.95ms
NorBERT3_{xs}	0.85	96.98	96.36	3.34ms
NorBERT3_{xs}	0.90	97.34	96.79	3.82ms
NorBERT3_{xs}	0.95	97.60	97.11	4.46ms
NorBERT3_{small}	0.25	94.04	92.85	0.99ms
NorBERT3_{small}	0.5	95.18	94.20	2.04ms
NorBERT3_{small}	0.75	96.93	96.28	4.23ms
NorBERT3_{small}	0.80	97.13	96.52	4.22ms
NorBERT3_{small}	0.85	97.24	96.67	4.59ms
NorBERT3_{small}	0.90	97.56	97.04	5.23ms
NorBERT3_{small}	0.95	97.95	97.52	6.22ms
NorBERT3_{base}	0.25	94.03	92.84	1.25ms
NorBERT3_{base}	0.5	95.09	94.10	3.98ms
NorBERT3_{base}	0.75	96.76	96.07	7.03ms
NorBERT3_{base}	0.80	96.95	96.31	7.53ms
NorBERT3_{base}	0.85	97.01	96.39	8.32ms
NorBERT3_{base}	0.90	97.23	96.66	9.76ms
NorBERT3_{base}	0.95	97.61	97.11	11.46ms

Table 7: Full list of hybrid NorBERT3 models with different confidence score thresholds, macro F1 and MCC on the test set and inference speed (ms/sentence)

Successes and Downfalls: Error Analysis of Scandinavian Language Identification

Victoria Handford, Marthe Midtgaard, Solveig Helene Willoch

Abstract

Language identification is a fundamental task in natural language processing and is often one of the first steps of processing multilingual data for use in further downstream tasks. While existing models perform strongly on high-resource languages like English, we observe a relatively poor performance for similar languages, for example Scandinavian languages. This paper presents a foundational exploration of the performance of selected existing language identification models on Scandinavian languages, highlighting some strengths and weaknesses. We compare these methods to a fine-tuned XLM-R and observe that we can achieve comparable performance using the No Language Left Behind (NLLB) model, thus saving time and resources needed to fine-tune large languages models for this task. We hope that our findings provide insights into Scandinavian language identification and where the existing models fail, in hope that this can help improve future models.

1 Introduction

Language identification (LID) is regarded as a solved task by many NLP researchers, but this is not necessarily the case for many world languages. For these languages, language identifiers either do not exist or perform relatively poorly. This is a major hurdle for developing NLP tools for them since language identification is essential for processing any large-scale multi-lingual data. For instance, if you train a Swedish language model, it learns from texts identified as Swedish. Similarly, a good Nynorsk machine translation model should produce texts identified as Nynorsk. The performance of these large and complex models is limited by the precision of the language identification tool used to curate the data in order to have a quality foundational training corpus.

Existing language identification models perform relatively poor when it comes to distinguishing

between the Scandinavian languages Danish, Norwegian and Swedish, as they are similar languages. The Scandinavian languages share common history rooted in the Viking Age, during which Old Norse was spoken across the region¹. This later evolved into the distinct Scandinavian languages, yet they retain similar linguistic features across vocabulary, grammar, and pronunciation. This contributes to the languages' mutual intelligibility today, posing a challenge for accurate language identification. Norwegian presents additional challenges as it is divided into two written forms, Bokmål and Nynorsk, both representing the same oral language.

In this paper we look at language identification models for Norwegian Bokmål, Norwegian Nynorsk, Danish, and Swedish. We consider existing methods – No Language Left Behind (NLLB), OpenLID, FastText, CLD3 – for a baseline and compare the results of the best performing model to a XLM-R model fine-tuned for Scandinavian LID. Additionally, we include a random baseline for comparison.

The sections of this paper discuss key aspects of our research. Section 2 outlines previous research in the field of LID. Section 3 introduces the datasets we used for training (when needed) and evaluating the models. Section 4 identifies the metrics we use for evaluation and justifies our choices. Section 5 presents our baseline models, and Section 6 explains the process of fine-tuning the large language model. In Section 7, we examine the results of model performance and identify main case categories into which errors can be clustered. Finally, Section 8 rounds off our research with a discussion of our findings, the limitations of our work, and suggestions for future research.

We aim to shed light on areas where the models perform well or poorly through a dedicated error analysis, identifying cases where the models fail

¹https://snl.no/nordiske_språk

to classify correctly, seeing which of the cases are the most difficult, and determining whether the mistakes are obvious to humans. In doing so, we hope to highlight the importance of quality training data and identify cases in that data where existing models tend to struggle to encourage creating and using more curated datasets.

2 Related Work

Research into language identification has a rich and long history, employing a diverse array of methods (Jauhainen et al.). Recent work in automatic LID has leveraged both traditional machine learning techniques and more advanced deep learning techniques. Although language identification for high-resource languages has become highly effective and is no longer a primary research focus, the challenge of distinguishing between similar languages continues to draw significant interest. This focus is exemplified by initiatives like the DSL (Discriminating between Similar Languages) shared task, which has been held since 2014 (Zampieri et al., 2014).

A recent study has explored various methods to discriminate between Nordic languages in the LID task (Haas and Derczynski, 2020). Linear machine learning models, FastText² and neural networks, including convolutional neural networks (CNNs) were evaluated. The study found that FastText’s performance was comparable to that of the CNN, which was the best-performing model, yet FastText is more efficient both during training and inference.

3 Data

For our experiments, we used the gold training and validation sets from the Universal Dependencies treebanks (Nivre et al., 2017), where the validation sets were used as the held-out test data and for error analysis. Each instance in the dataset is a sentence with a single label identifying the sentence’s language. Table 1 shows how the data is split. The following shows the first instance in the training data:

```
{
  'text': 'For no skal vi drive
          ut desse demonane.',
  'languages': "['nn']"
}
```

For the purpose of our experiments, we only consider the data with single labels. We acknowledge

²<https://fasttext.cc>

that some instances can be acceptable in multiple languages, but for simplicity’s sake, we consider only single-labeled data (more on this in Section 8). Additionally, it is important to note that the datasets are unbalanced, that is to say, there are more examples of Bokmål and Nynorsk in the datasets than Other, Danish, or Swedish, as shown in Table 1.

Language	Label	Training	Evaluation
Bokmål	nb	15435	2392
Nynorsk	nn	14029	1859
Other	other	8360	1116
Danish	da	4376	563
Swedish	sv	4278	503
Total		46478	6433

Table 1: The number of instances in each dataset by language. The training data was used for training, whereas the validation set was used as a held-out test set and for error analysis.

4 Metrics

For evaluating the models, we look at three metrics, namely accuracy, macro F1 score, and Matthews’ correlation coefficient (MCC)³. The first metric, accuracy, tells us how close the models’ predictions are to the target labels. The macro F1 score is another quantification of performance that combines the precision and recall scores. Thirdly, the Matthews’ correlation coefficient is another representation of the quality of binary classification.

We calculate each method *loosely*, meaning that the target value is only one language label, and the model predicts only one language label. The metrics are calculated with an all-or-nothing approach, that is to say, either the prediction is correct or incorrect, even though some instances are acceptable in multiple languages as we address later in the paper. In addition, we acknowledge that the number of examples for each language is not the same. While the datasets are unbalanced, we find the choice to use macro scoring over weighted scoring suitable due to the fact that the most similar examples are those of Bokmål and Nynorsk, so we need more examples to get a representative result of model performance.

5 Baselines

Initially, we calculate the results of randomly predicted languages as a lower level benchmark. Next,

³<https://lightning.ai/docs/torchmetrics/stable/all-metrics.html>

we use several different linear models for our baselines, the first of which is the original FastText model for language identification (Joulin et al., 2016). Then, we evaluate the next two models, OpenLID (Burchell et al., 2023) and NLLB (NLLB Team et al., 2022), which train a linear classifier on top of the FastText model using curated datasets and linguistic input. As a final baseline, we evaluate the CLD3 model⁴ which is a neural network that uses character-based n-grams as features.

By using different types of baseline architectures, we hope to gain insight into what improves model performance and highlight decisions which can negatively affect results. For example, the OpenLID and NLLB language identifiers further refine the performance of the FastText model with the added classifier. If this is a successful approach, we will see that the OpenLID and NLLB language identifiers will outperform the FastText model. On the other hand, the CLD3 model only has one class for the Norwegian language and does not distinguish between Norwegian Bokmål and Norwegian Nynorsk. Therefore, we had manually decided whether instances identified as Norwegian would default to Bokmål or Nynorsk, and decided to default to Bokmål as it makes up a larger portion of the dataset. However, this means that no instance is identified as Nynorsk by the CLD3 model, disrupting the performance results.

6 Fine-Tuned Language Model (FTLM)

We fine-tuned the XLM-R base model (Conneau et al., 2019) on the gold training dataset. To save time and resources, we used the procedure and hyperparameters used in (Luca Papariello, 2024) as a starting point and adapted it to our specific task (Table 2). In their experiments, they fine-tuned the base XLM-R model to the LID task and were able to predict across 20 languages with an average accuracy of 99.6 percent. We acknowledge that, by opting out of a more in-depth hyperparameter search, the performance of the fine-tuned classification model could be improved. However we chose to prioritize error analysis over hyperparameter optimization.

Additionally, we decided to limit inputs to a length of only 128 characters. This is a naive simplification and we acknowledge that we lose some information by cutting off examples early. For instance, Example 1 is a sentence in Nynorsk where

⁴<https://github.com/google/clD3/>

Learning Rate	Train Batch	Eval. Batch	Epochs
2e-05	64	128	2

Table 2: Hyperparameters for fine-tuning the base XLM-R model. The fine-tuning process took around 12 minutes on an NVIDIA L4 GPU.

the colored words are indicators of Nynorsk rather than Bokmål, and the character cutoff is marked by [128] and a line break. We see that there are a total of 12 indicators of Nynorsk, but only 3 occur within the first 128 character section. Therefore, by expanding the input window, we can provide more information to the model and potentially improve performance. However, we see that most of the data consists of fewer than 128 characters (Q3: 118 characters, *see Appendix 4 for complete statistics*), so examples like Example 1 are not very frequent, and we decided to prioritize simplifying computation over potentially optimizing input length.

- (1) Ei slik løysing har etter måten solid tilslutning gjennom dei merknader og framlegg som er komme i samband med det språkpolitisk [128]
e grunnlagsarbeidet som er gjort dei siste åra, og er elles ein naturleg konsekvens av at regjeringa med denne meldinga tek til orde for ein ny og heilskapleg språkpolitikk som skal omfatta den totale språksituasjonen i landet.

7 Results

Model	Accuracy	F1	MCC	Runtime
FTLM	0.9541	0.9567	0.9376	~ 3.5 m
NLLB	0.9307	0.9305	0.9154	0.84 s
OpenLID	0.9178	0.9176	0.9004	0.84 s
FastText	0.7298	0.7424	0.7205	0.23 s
CLD3	0.6370	0.6392	0.6069	0.64 s
Random	0.1940	0.1773	0.2475	0.01 s

Table 3: Loose metrics for baseline performance of existing methods on development sets. Runtime is represented by the CPU inference time in either seconds (s) or minutes (m).

We evaluated each of the baseline algorithms using the provided evaluation script⁵, and the overall model performance results are shown in Table 3

⁵evaluation.py

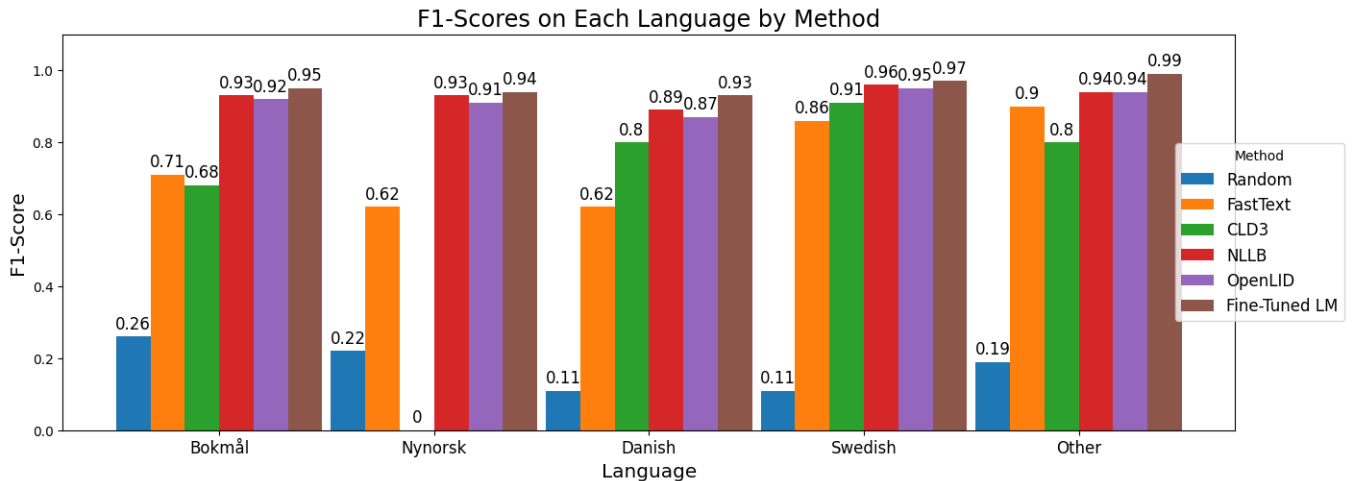


Figure 1: F1-Score received by each of the models for each of the languages. Since the CLD3 model does not include separate labels for Norwegian, it receives an F1-score of 0 for Nynorsk, and the Bokmål results are skewed.

and Figure 1. The results also includes the performance of the FTLM. We see that, as expected, all models perform better than the random baseline, and the NLLB language identifier performs the best of all of the baselines. When looking at the inference times of the models, the runtime increase, albeit slightly, as the performance increases. Notably, the fine-tuned language model before the best, but at 250x the runtime of the NLLB language identifier achieving only around a 2 percent bump in performance.

As introduced in Section 5, we see the impact of different approaches on the results. The CLD3 model does not perform well on Norwegian emphasizing the necessity of separating Nynorsk and Bokmål in LID tasks, as it performs better than the FastText on Swedish and Danish. We see also that using a curated dataset to fine-tune a classifier on top of the FastText model is successful, as shown by the NLLB language identifier and OpenLID model results.

For our analysis, we look at all models from a more surface level perspective and look at general statistics in Section 7.1. Then, we dive deeper into the best performing models and attempt to identify problem areas that give insight into what the models need to improve on in Section 7.2.

7.1 General Statistics

We see that the best performing existing LID models (NLLB and OpenLID) achieve F1-scores on each language which are generally within 0.05 of the performance of the fine-tuned language model.

When looking at the misclassified examples, we

find that they tend to be shorter (i.e. consist of fewer characters) than the overall distribution of lengths in the dataset. Figure 2 shows a density plot of the character length of misclassified examples by model compared to the distribution of lengths in the entire dataset. We see that there is a higher density of misclassified examples shorter than 40 characters for the NLLB, OpenLID and FTLM models than in the dataset as a whole.

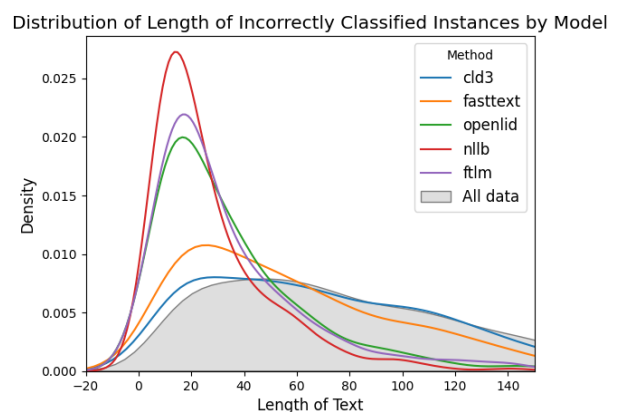


Figure 2: The distribution of the character length of incorrectly classified instances by model compared to the the distribution of lengths in the entire dataset. This figure has a reduced domain for clarity. Appendix 6 shows the entirety of the density plot.

While the distribution of the lengths of examples misclassified by the CLD3 model – and to some extent, the FastText model – is fairly close to that of the entire dataset, we attribute this is caused by the overall poor performance of the model rather than that the model handles shorter examples particularly well. Ultimately, all models face difficulties

when predicting short ambiguous sentences due to the lack of clear context, something even humans find difficult. We continue the discussion of ambiguity in Section 7.2.

Next, we looked at what the models predicted when they misclassified an instance (see Figure 3). We see that NLLB language identifier and the OpenLID model are similar when it comes to the amount to misclassifications when the model predicts Danish, Bokmål and Swedish, but it is interesting to note that OpenLID predicts Nynorsk more often than NLLB when it classifies incorrectly. On the other hand, NLLB predicts Other incorrectly more often than the other two models shown. However, it is important to note that NLLB was trained to identify many more languages than just the four we examine and the dataset is not perfect (see *Potentially Problematic Gold Labels* in Section 7.2) so we see that even though our metric punishes the model for predicting Other, it is not always technically incorrect. Additionally, we see that the fine-tuned language models predicts Bokmål incorrectly much more than the other two models and does not predict Other nearly as frequently. However, it is unclear whether this is statistically significant because it was fine-tuned on an unbalanced dataset which favored Bokmål and had fewer examples of other languages.

7.2 Case Categories

We chose to further analyze the two best models, namely the fine-tuned model and the NLLB model, and look in depth at where the models failed to classify correctly. In our analysis, we were able to identify four misclassification categories relating to ambiguity: input from Non-Scandinavian Languages, potentially problematic gold labels, named entities, and ambiguity because of language similarity.

When it comes to determining the cause of misclassification, we find that the length of the sentence carries more weight than the following cases of ambiguity. If an input consists of fewer characters, it is clearer that the lack of information leads to misclassification and overrides the influence of the named cases. For longer sentences, it becomes easier to point out which of the four cases plays the biggest part for the misclassification.

Input From Non-Scandinavian Languages

Misclassification often occurs when the presence of words in non-Scandinavian languages outweighs

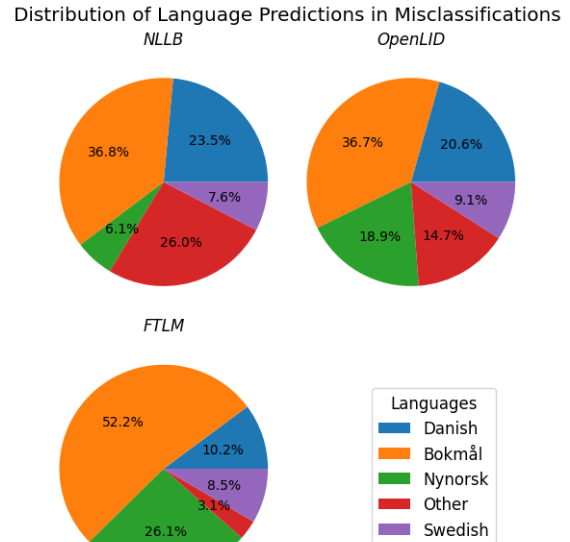


Figure 3: When the models predict a language incorrectly, this is the distribution of the labels predicted.

the linguistic indicators that identify the text as Scandinavian. This could be the case when test sentences contain non-Scandinavian references and loanwords.

When comparing misclassifications due to the inclusion of non-Scandinavian words, the NLLB model exhibits a tendency to misclassify clearly non-Scandinavian sentences as Scandinavian, while the fine-tuned model demonstrates far fewer instances of that sort, likely due to the reduced size of languages seen during training for the fine-tuned model. In the NLLB model’s training data, instances with foreign scripts inside sentences of another language were found (Costa-jussà et al., 2022). Despite attempts to filter out such sentences, some may have been missed, leading to misclassification of test sentences due to incorrectly learned language representations (see Example 2).

- (2) et venerunt et impleverunt ambas naviculas
ita ut mergerentur (pred: nb, gold: other)

Additionally, the NLLB model frequently misclassifies sentences with loanwords and non-Scandinavian references. In such cases, sentences are often misclassified as Other when the number of Non-Scandinavian words surpasses or equals the count of Scandinavian words (see Example 3). This stands in contrast to human evaluation, where

such instances are typically correctly classified. On the other hand, the presence of loanwords and Non-Scandinavian references has an overall diminished impact on the fine-tuned model’s decision-making process. This is due to the transformer-based architecture of the fine-tuned model that creates a contextualized representation better identifying language-specific features crucial for accurate classification.

(3) Ein avocado (gold: nn, pred: other)

However, both models still misclassify when sentences only include loanwords and non-Scandinavian references and words, highlighting the inherent challenge of accurately classifying sentences acceptable in multiple languages. Such sentences are typically short, lacking linguistic cues and context indicative of the correct language. Consequently, they can fit into both the gold and predicted languages, introducing ambiguity, as seen in Examples 4 and 5 from the fine-tuned model. These sentences would probably also be misclassified by humans since they do not have any words that are specifically Scandinavian, and the same loanwords can appear in all the languages.

(4) Voila! (pred: other, gold: nn)

(5) «Kids Wanna Rock» (pred: nb, gold: other)

Potentially Problematic Gold Labels Generally, language identification works better on the documents level and is more difficult at the sentence level due to less context. This leads to instances where the instance itself seems to be in one language, but is included in text that is labeled as another language, e.g. using an English idiom in a Bokmål paragraph. We see that models predict correctly, i.e. label the instance as other, even though they are punished because the gold label is nb.

We find that both NLLB and the fine-tuned model struggles with these problematic gold labels. It is not surprising that the models misclassified these instances since the sentences themselves are not in any of the Scandinavian languages when considered in a stand-alone context. We found a few instances (see Example 6 and 7) where the example seems to be in English and the model predicted other, while the sentence was labeled with one of the Scandinavian languages in the gold dataset. Interestingly, we also found a Non-English sentence that have a problematic gold label (see Example 8).

This sentence is in fact not a Nynorsk sentence, but a Bokmål one.

(6) « ... where do we go when we die? (pred: other, gold: nb)

(7) What lies beyond, and what lay before? (pred: other, gold: nb)

(8) Det forutsetter selvsagt at komiteene engasjerer seg på saker som krever det. (pred: nb, gold: nn)

The reason for these problematic gold labels are because the sentences appear in documents of the respective language and has therefore been labeled as such. The misclassification of such sentences is therefore not because of the model, but the out-of-context labels themselves, resulting in a negative effect for the metrics score the models are given since they are seen as mistakes when that is not the fact.

Named Entities Named entities have meaning that carries across language boundaries and may contain character sequences that lack language-specific patterns and are common across multiple languages, which could lead to misclassification. For example, Example 9 is misclassified by both the NLLB and FTLM model as Danish, even though the word *veske* (purse) is not a Danish word. Both the NLLB model and the fine-tuned model face challenges with sentences including named entities, particularly in short sentences. However, the fine-tuned model typically predicts a Scandinavian language, whereas the NLLB model tends to predict Other (see Example 11 and 10, respectively).

(9) Jeg elsker min Chanelveske. (pred: da, gold: nb)

(10) Leste Karl Marx (pred: other, gold: nb)

(11) (Foto: Bent Rej) (pred: nb, gold: da)

The tendency for the fine-tuned model to wrongly predict a Scandinavian language may stem from the bias towards Scandinavian languages in the training data, coupled with the inherent similarity among Scandinavian names, which are less frequently encountered in the Other category of the training data. More specifically, sentences with named entities tend to be misclassified as Bokmål, possibly due to the dominance of Bokmål in the training data, leading to a greater exposure

of named entities in the Bokmål training data. Consequently, even if a sentence exhibits linguistic features indicative of another Scandinavian language, the model’s preference towards Bokmål sometimes prevails (see Example 12).

(12) Jytte Hilden og Lone Dybkjær er barndomsveninder, og Poul Nyrup Rasmussen og Mogens Lykketoft har været venner i mere end 25 år. (pred: nb, gold: da)

Finally, when a few extra words are included, the fine-tuned model typically demonstrates greater proficiency in predicting the language than the NLLB model does. This advantage may stem from the model’s contextualized understanding, enabling it to identify and effectively utilize the few linguistic cues for more accurate language prediction. In contrast, the NLLB model relies solely on statistical probabilities from character sequences. This makes it difficult to prioritize words for prediction due to the model’s lack of contextual understanding. Thus, while both models may encounter difficulties with short sentences containing named entities, the fine-tuned model’s architecture allows for better handling of such cases.

Ambiguity Because of Language Similarity Scandinavian languages have many features in common, such as word forms and grammar. We find that there are two instances where the misclassification tends to happen because of this similarity. The first is sentences that are misclassified because of ambiguity where they are correct for multiple languages, but the data has only one gold label. If the model predicts one of the languages that is not the gold label, it will be seen as incorrect, even if it is not necessarily so. The second instance is sentences where there are only a few words that are uniquely language specific, but the rest are words that are found in multiple languages.

From Figure 4 and Figure 5, both models struggle the most with predicting Nynorsk, and most of the wrongly predicted sentences were predicted to be Bokmål. For many of these sentences, it is the presence of one or two language specific words that identify the sentence as one of the languages. For instance, Example 13 is incorrectly predicted as Bokmål, and the verb *vere* is the only indicator that it is in Nynorsk, as well as Example 14 where *ministerbilerne* and *tobaksrygning* are the only indicators that it is in Danish. From the two figures we also see a trend for both models where Bokmål

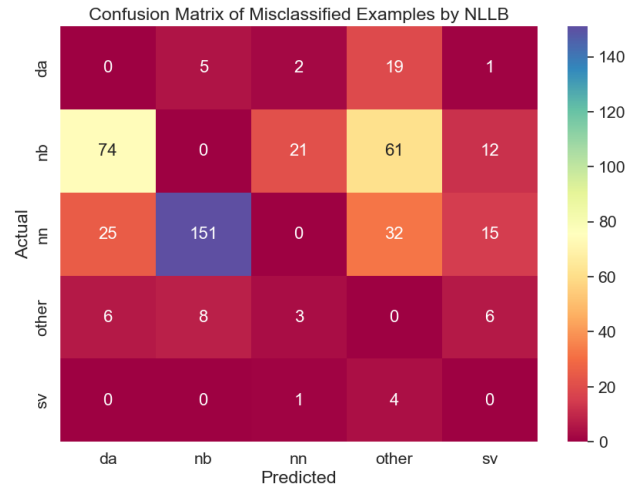


Figure 4: Number of incorrectly predicted languages per language for NLLB

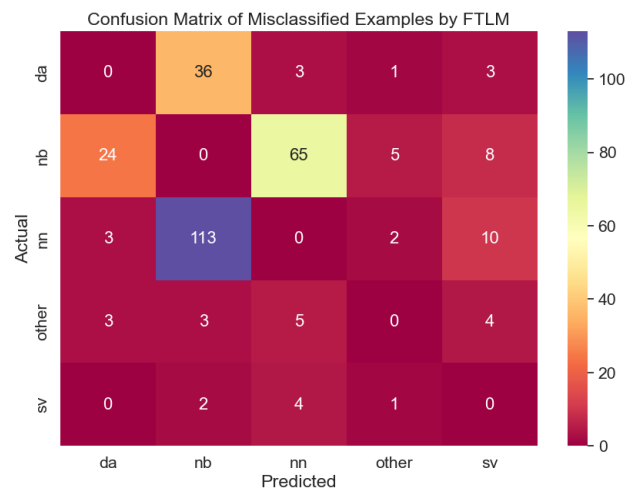


Figure 5: Number of incorrectly predicted languages per language for fine-tuned LM

sentences are misclassified as Danish. The figures also show that, in general, the fine-tuned model has the most difficulty distinguishing between Nynorsk, Bokmål, and Danish. It often misclassifies a sentence in one of these languages as being in one of the other two. NLLB, also struggles with Nynorsk and Bokmål, but the misclassifications are more spread out between all the languages. We can also see that NLLB has a higher tendency to misclassify a sentence as Other, than the fine-tuned model.

(13) Distrikt norge er kjent for å vere bakpå og intolerant. (pred: nb, gold: nn)

(14) Med andre ord: tobaksrygning er forbudt i ministerbilerne! (pred: nb, gold: da)

There are a number of ambiguous sentences that are acceptable in multiple Scandinavian languages because of their shared orthography, which makes them difficult to classify. Both models tend to struggle with these because of the similarity and lack of language identifiable words. Example 15 is allowed in both Danish and Bokmål, Example 16 is allowed in Nynorsk, Bokmål, and Danish, Example 17 is allowed in all the Scandinavian languages, while Example 18 is allowed in both Swedish and Nynorsk. In our analysis, we found that these difficulties appear most often between Bokmål and Danish, and Nynorsk and Bokmål, given their close orthographic proximity.

(15) Det er et 'jeg' der, det er et 'vi' der og det er et 'de' der. (pred: da, gold: nb)

(16) Metoden er brutal. (pred: da, gold: nn)

(17) (Privat foto) (pred: sv, gold: nn)

(18) Det handlar om systemet (pred: nn, gold: sv)

It comes as no surprise that the models struggle more between Nynorsk and Bokmål than between the other languages considering they are written forms of the same oral language. In addition, there have been a number of changes in the orthographic standardization for Nynorsk ([Språkrådet](https://www.sprakradet.no/globalassets/vi-og-vart/publikasjoner/ny-nynorskrettskriving.pdf)) where some of the argumentation for the changes are to make it easier for the speakers of Norwegian to read and write both written languages. Two example from the last change that happened in 2012⁶ was that it is now allowed to write *skole* in addition to *skule*, and *enten* and *anten* is equally acceptable to write. While the changes might make it easier for the speakers of Norwegian by making the two written forms more similar to each other, this is the exact thing which makes it more difficult for models to learn the difference, and thereby predicting which of the two written forms is used.

8 Discussions

Our research examined the successes and downfalls of existing language identification models on Scandinavian languages. Of the baseline algorithms we looked at, NLLB performs the best and achieves performance comparable to the results of our fine-tuned language model. This suggests that we do not

⁶<https://www.sprakradet.no/globalassets/vi-og-vart/publikasjoner/ny-nynorskrettskriving.pdf>

necessarily need to invest resources in fine-tuning models, but rather that we can use existing models to achieve comparable results.

In our error analysis, we identified several clusters of misclassification cases where models fail to disambiguate Scandinavian languages. Firstly, short examples likely do not provide enough information for the model to identify the language. Next, we looked at four cases of ambiguity in the incorrectly labeled instances to highlight potential problem areas. The model may classify the instance incorrectly if multiple languages are represented in the instance. When a document includes loanwords or Non-Scandinavian language, it can confuse the model or potentially have a misleading gold label. Language-agnostic forms, like names, places, dates, or citations, are a further hindrance, as they do not change. Examples with shared written forms across languages were a source of problems for the models, although we mention that this is also the case for humans as well since there are multiple correct answers.

8.1 Limitations

Due to practical and resources restraints, we acknowledge the limitations of our work:

- The performance by our fine-tuned model could be improved with a more in-depth hyperparameter search and is limited by a fixed context window.
- The statistics are based on all or nothing classification, when, as we have seen, Scandinavian Language identification can be ambiguous and not always suited for single-label classification.
- The potentially problematic gold labels influence the models ability to learn an accurate representation of the language.

8.2 Future Work

We see that the majority of errors in our best performing models stem from ambiguity in the examples, where there is not a single correct label. This suggests that binary or multi-class classification is not sufficient enough for this task, and multi-label classification is required for further improvements. We encourage future work in how we can allow multi-label classification in cases where a sentence is acceptable in multiple Scandinavian languages or ambiguous due to lack of context.

Additionally, we hope that our work will motivate continued research into curated datasets, as (Burchell et al., 2023) and (NLLB Team et al., 2022) have done. We have shown that adding a curated dataset improves performance and highlighted problem cases and hope that they can be used to further refine training data to improve model performance. As we saw with the presence of foreign texts, mostly English, in the Scandinavian datasets, we encourage a finer-grain refinement of datasets to prevent misleading gold labels that could prohibit a model from learning an accurate representation of the language.

References

- Laurie Burchell, Alexandra Birch, Nikolay Bogoychev, and Kenneth Heafield. 2023. [An open dataset and model for language identification](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 865–879, Toronto, Canada. Association for Computational Linguistics.
- 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](#). *CoRR*, abs/1911.02116.
- Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.
- René Haas and Leon Derczynski. 2020. Discriminating between similar nordic languages. *arXiv preprint arXiv:2012.06431*.
- Tommi Jauhiainen, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. *Automatic Language Identification in Texts*. Springer Nature. Google-Books-ID: n9PrEAAAQBAJ.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.
- Luca Papariello. 2024. [xlm-roberta-base-language-detection \(revision 9865598\)](#).
- Joakim Nivre, Daniel Zeman, Filip Ginter, and Francis Tyers. 2017. Universal dependencies. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*.
- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia-Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation.
- Språkrådet. [Rettskrivningsvedtak](#). Accessed: 2024-05-08.
- Marcos Zampieri, Liling Tan, Nikola Ljubešić, and Jörg Tiedemann. 2014. [A report on the DSL shared task 2014](#). In *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*, pages 58–67, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

A Appendix

method	mean_len	min_len	Q1	median_len	Q3	max_len
nllb	27.61	1.0	11.0	20.0	36.0	182.0
ftlm	33.59	1.0	14.0	23.0	43.5	145.0
openlid	35.72	1.0	14.0	26.0	47.0	210.0
fasttext	64.28	1.0	25.25	50.0	89.0	437.0
cld3	77.76	1.0	34.0	67.0	108.0	437.0
all_data	85.97	1.0	41.0	73.0	118.0	664.0

Table 4: Statistics about the distribution of the character length of the incorrectly classified instances by model. "all_data" shows the distribution of text length in the entire evaluation dataset.

Distribution of Length of Incorrectly Classified Instances by Model

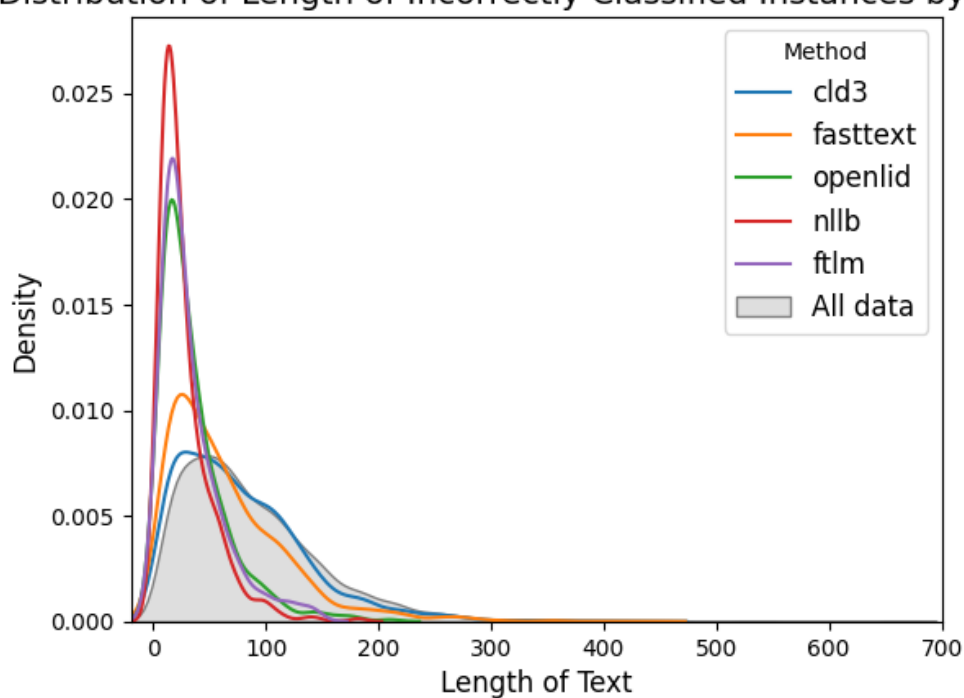


Figure 6: Density plot of the lengths of misclassified examples compared to the lengths of all examples in the dataset.

Transformers do (not) outperform FastText on Scandinavian language identification

Mariia Fedorova

University of Oslo, Norway

mariiaf@ifi.uio.no

Abstract

This paper presents a fast and accurate model for Scandinavian language identification on short news texts. We compare transformer-based models with more lightweight ones (based primarily on static word embeddings) both in CPU inference speed and accuracy and try to find the balance between these two metrics using quantization and knowledge distillation. Our best model infers 9 times faster than the best baseline while preserving its F1-score¹.

1 Introduction

Although transformer-based large language models have proved to gain state-of-the-art results on many text classification tasks, including benchmarks for Scandinavian languages (Snæbjarnarson et al., 2023), their practical usefulness for some real-world tasks is questionable. Language identification is a first step in many text processing pipelines and is required to be not only accurate, but also fast. In this paper we compare different models and optimization techniques for their performance on the task of identifying, whether a text is written in Danish, Norwegian Bokmål, Norwegian Nynorsk, Swedish or other language.

The task of Scandinavian language identification is more complicated than a regular language identification task, since three of the four aforementioned languages share the same alphabet and all four languages have the same ancestry, which results in the fact that whole sentences can be written in the same way, not only separate words. Creation of multilabeled corpora for solving this problem requires costly manual verification.

2 Related work

Language identification as a whole and Scandinavian language identification in particular are under-

explored fields.

A default baseline solution for text classification tasks are bag of word (or subword) vectors, for instance, CLD3² is a feed forward neural network that uses them as input features. This model treats both Bokmål and Nynorsk as the same class Norwegian. Although such a model is simple to train and efficient to predict, it may be not robust towards unknown words or word forms.

Some models, solving the task efficiently and accurately, are FastText (Joulin et al., 2017), NLLB (Grave et al., 2018), OpenLID (Burchell et al., 2023), GlotLID (Kargaran et al., 2023). They all use FastText word vectors (Bojanowski et al., 2017) and make a difference between Norwegian Bokmål and Nynorsk. The Kargaran et al. (2023)’s model³ is especially interesting because it includes low-resource languages along with high-resource ones, which results in 2155 classes in total, while the three other include 176, 207 and 217 languages only.

We use all these models as our baselines, the results will be discussed in the next sections.

3 Data

We report only the results obtained with the gold training/validation data from Universal Dependencies treebanks (Nivre et al., 2017). These datasets are not disambiguated (a sentence labeled as Nynorsk is in Nynorsk, but it could also be a valid Bokmål sentence).

The test data are from the same treebanks, but manually annotated in a multilabeled way. For example, some samples are just named entities (e.g. ‘Marie’) labeled with all four languages.

Number of samples (sentences) across the datasets and languages is shown in Table 1. The dataset is disbalanced towards Norwegians. Al-

¹Code is available at: <https://github.com/mariiaf/in9550-exam>

²<https://github.com/google/cld3>

³<https://huggingface.co/cis-lmu/glotlid>

Language	Gold train	Gold dev	Test
Bokmål	15 435	2 392	1 939
Danish	4 376	563	565
Nynorsk	14 029	1 859	1 511
Swedish	4 278	503	1 219
Other	8 805	1 165	1 742

Table 1: Number of samples (sentences) across the datasets and languages.

though this may be useful for Norwegian-focused downstream tasks, this may be not a real world distribution (e.g. the Norwegian Bokmål Wikipedia contains about 630 000 articles⁴, while the Swedish one contains about 2 585 000 articles⁵). For this reason, we did not experimented with class weights for loss functions while training our models.

4 Experiments

4.1 Evaluation metrics

We evaluate both single-label and multilabel formulations of the task. The first one is measured by ‘loose’ metrics: if the predicted label is among the gold labels, it is correct. The ‘strict’ metrics consider to be correct only those sets of labels, that exactly match with the gold ones. We report both loose and strict accuracy and macro F1 for each model.

4.2 Baselines

Table 2 shows the baseline results and CPU inference time (it is approximate, but allows to divide the methods into ‘slow’ and ‘fast’). The inference time is reported for the whole test set, also including the preprocessing (tokenization etc.) steps.

For the best-performing FastText methods we experimented with reducing the number of out of vocabulary words (OOV) by removing punctuation and lowercasing (entries with * in 2 show these results).

4.3 Single-label vs. multilabel training

First, we had to choose between single-label and multilabel formulations of the task. For this purpose, we fine-tuned a lightweight (15 M parameters) NorBERT3-XS model (Samuel et al., 2023) and trained it with both regular Cross Entropy and Binary Cross Entropy loss functions. The results

⁴<https://no.wikipedia.org/wiki/Portal:Forside>

⁵<https://sv.wikipedia.org/wiki/Portal:Huvudsida>

turned out to be worse for the multilabel setup (see Table 3), which was expected, since the training set contained ambiguous examples labeled by a single label. Manual inspection of class probabilities for random ambiguous samples showed, that all true probabilities are higher, than the wrong ones, but those of a more frequent class in the data are much higher (it was also proved by a larger difference between accuracy and F1 for the multilabel model than for other models). A typical example is the sentence ‘*Byen er totalt mørk*’ (‘*The city is totally dark*’), belonging to Bokmål, Nynorsk and Danish, which got the probability distribution of approximately 0.99 Bokmål, 0.012 Nynorsk, 0.011 Danish, 0.0011 Swedish and 0.0016 other. Thus, a multilabel model trained on single-label data behaves like a single-label model trained on a noisy corpus.

It was also impossible to establish a threshold for the class probabilities, since the difference among them varied from sample to sample. For these reasons, we left the creation of a multilabeled training set for future work and trained the rest of our models for a single-label task. However, we still report both loose and strict metrics, in order not to oversee a situation when excessive optimization of a loose metric caused the strict ones to sink.

4.4 Bertology

Model comparison. After making it clear with the task formulation, we had to choose a language model. BERT (Devlin et al., 2019) is nowadays a default sequence classification solution based on a transformer encoder (Vaswani et al., 2017) language model. We compared multilingual, Scandinavian and Norwegian BERTs with different number of parameters in order to choose a model to focus on for further optimizations. Since performance on CPU is important for us, we began with smaller models and moved on to larger ones.

The Scandinavian model we use is ScandiBERT⁶ (Snæbjarnarson et al., 2023) and, speaking precisely, is neither a BERT nor a pure Scandinavian model, but an XLM-RoBERTa-base (Conneau et al., 2019) further trained on Icelandic, Danish, Norwegian, Swedish and Faroese.

The results are presented in the table 4. While an ‘XS’-sized model performed below the best baseline (GlotLID), all ‘S’- and more sized models outperform it. One can note that the difference between ‘S’ and ‘base’ sizes does not contribute

⁶<https://huggingface.co/vesteinn/ScandiBERT>

System	Accuracy, loose, %	F1, loose, %	Accuracy, strict, %, %	F1, strict, %	CPU
CLD3	72.36	65.42	70.50	65.03	0.78
OpenLID	93.51	92.39	90.54	90.48	0.48
FastText	79.69	76.74	77.58	78.89	0.12
NLLB	94.41	93.21	91.71	91.54	1.89
NLLB*	94.93	94.15	92.07	92.35	1.04
GlottLID	96.30	95.17	93.53	93.38	4.75
GlottLID*	96.09	94.95	93.38	93.28	4.66
random	22.05	21.04	20.50	20.78	0.02

Table 2: Baseline metrics

System	Acc, loose, %	F1, loose, %	Acc, strict, %	F1, strict, %	CPU
norbert-xs, single	92.70	91.29	89.28	88.68	125.79
norbert-xs, multi	90.34	87.74	87.16	85.61	141.70

Table 3: Metrics of the models trained for single- or multilabel classification. F1 here and in all other tables is always macro averaged. CPU inference time is always in seconds.

much to the quality, while the inference speed becomes twice as worse. Moreover, the DistilBERT multilingual cased model⁷ (Sanh et al., 2019) performs better than the model it was distilled from⁸ (Devlin et al., 2019).

Another interesting result is that the Norwegian models^{9,10} outperform the multilingual models.

Inference speed optimization. Since inference time of the fine-tuned BERT models is more than 40 times as large as the GlotLID’s one, while their evaluation metrics are only about 5 point better, we had a look common technique of inference speed optimization. First, we export a model into ONNX¹¹ runtime, which reduces the running time of an ‘S’-sized model up to less than a minute without loss in its accuracy. Then we apply a quantization into int8 datatype, which further decreases model size and inference time with affecting performance only a little. The exact results are presented in the table 5.

4.5 Knowledge distillation

Although the inference time of 43 seconds on 6976 samples may be satisfactory for some tasks, even a quantized BERT is still much less efficient than GlotLID. For this reason, we tried to train a tiny

model that would profit from both GlotLID and NorBERT-base, following the approach from (Hinton et al., 2015): while training a light student model, we run prediction by both the student and a larger teacher model and calculate two losses. The first one is calculated as follows:

$$\frac{t * \sum(\log t - s)}{\text{len}(s) * (T^2)}$$

where s and t are student and teacher class probabilities in a batch, and T is a hyperparameter (we used 2). We will further call it Soft Target loss. The second one is a regular Cross Entropy loss of the student. Then both losses are multiplied by a weight number each. Both Soft Target loss weight and Cross Entropy loss weights are hyperparameters. We used 0.25 and 0.75 respectively¹². The sum of the two weighted losses is used for back-propagation.

We used a feed forward network with 1 hidden linear layer of size 64 and a ReLU activation function between it and the output linear layer. GlotLID sentence embeddings¹³ with a size of 256 served as an input. Table 6 shows the metrics both for the student model trained solely and with the teacher. Checkpoints trained with a teacher are saved with TorchScript¹⁴, for this reason they use less memory. Inference time includes also getting a sentence embedding from GlotLID (its size is 1.69GB). This

⁷<https://huggingface.co/distilbert/distilbert-base-multilingual-cased>

⁸<https://huggingface.co/google-bert/bert-base-multilingual-cased>

⁹<https://huggingface.co/lgt/norbert3-small>

¹⁰<https://huggingface.co/lgt/norbert3-base>

¹¹<https://github.com/onnx/onnx>

¹²https://pytorch.org/tutorials/beginner/knowledge_distillation_tutorial.html

¹³<https://github.com/cisnlp/GlottLID?tab=readme-ov-file#for-getting-sentence-vectors>

¹⁴<https://pytorch.org/docs/stable/jit.html>

System	N params, M	Acc, loose,	F1, loose	Acc, strict	F1, strict	CPU
DistilBERT	66	96.26	96.06	92.73	93.19	131.43
BERT-base	110	95.84	95.71	92.29	92.80	281.21
ScandiBERT	250	97.81	97.54	94.35	94.64	250.78
NorBERT-s	40	98.62	98.24	95.11	95.32	166.23
NorBERT-base	123	98.65	98.40	95.15	95.45	349.70

Table 4: Models with different sizes, NorBERT-s is for NorBERT 3 small. Accuracy and F1-score are given in percents.

System	Size, MB	Acc, loose	F1, loose	Acc, strict	F1, strict	CPU
NorBERT-s	187.7	98.62	98.24	95.11	95.32	166.23
NorBERT-s, onnx	179.3	98.62	98.24	95.11	95.32	58.95
NorBERT-s, onnx, QInt8	47.2	98.49	98.10	94.98	95.15	42.61

Table 5: ONNX optimization of NorBERT 3 small. Accuracy and F1-score are given in percents.

may explain why the inference time of both checkpoints is the same.

Although the knowledge distilled model does not achieve the teacher’s performance, it outperforms the solely trained student model and reaches the GlotLID performance (and even slightly outperforms its by F1 score) while being almost 5 times faster. Using a teacher also helps the student model to converge faster (we used early stopping for defining the number of training epochs).

This approach also leaves a lot of space for improvement, from hyperparameter search for the current loss to trying other knowledge distillation losses. It is also possible to train both the teacher and student models jointly. Thus, although the current result is not an outstanding one, the knowledge distillation approach deserves further development.

5 Error analysis

5.1 Tokenization

The finding that multilingual models perform worse for Scandinavian language identification than a Norwegian one is counterintuitive, especially given that we have an ‘other’ class including various languages of the world. One already reported problem with multilingual models applied to Scandinavian languages (Samuel and Charpentier, 2024) is that they tokenize the words in a way that does not reflect their morphology. We supposed that it could also cause problems in our task and compared how some samples are tokenized.

A typical example of much different tokenization that most likely caused an error is a Bokmål example ‘Ved å senke kvaliteten på det nye syke-

huset, vil driftskostnadene øke.’ DistilBERT erroneously classified it as Nynorsk, while NorBERT’s prediction was correct.

- **DistilBERT:** ‘[CLS]’, ‘Ved’, ‘å’, ‘sen’, ‘##ke’, ‘k’, ‘##vali’, ‘##teten’, ‘på’, ‘det’, ‘nye’, ‘sy’, ‘##ke’, ‘##huset’, ‘,’ ‘vil’, ‘drift’, ‘##sko’, ‘##st’, ‘##naden’, ‘##e’, ‘ø’, ‘##ke’, ‘,’ ‘[SEP]’
- **NorBERT3-base:** [‘[CLS]’, ‘ Ved’, ‘ å’, ‘ senke’, ‘ kvalitetet’, ‘ på’, ‘ det’, ‘ nye’, ‘ sykehuset’, ‘,’ ‘ vil’, ‘ driftskostnadene’, ‘ øke’, ‘,’ ‘[SEP]’]

In this example, DistilBERT’s tokenizer splits the words excessively and splits them not on morpheme borders as good tokenizers do. For instance, splitting of the word ‘kvaliteten’ (‘the quality’) does not make any sense and destroys the usefulness of subword tokenization. Moreover, we are cautiously suspicious that the resulting subwords can really occur both in Bokmål and Nynorsk.

One could argue that the previous example was classified correctly not because of better tokenization, but simply because a Norwegian model always outputs Norwegian. We have a counterexample for that, a Swedish sentence ‘En samling kalla fakta’. DistilBERT claimed it to be Nynorsk, while NorBERT3-base classified it correctly.

- **DistilBERT:** ‘[CLS]’, ‘En’, ‘samling’, ‘kalla’, ‘fakt’, ‘##a’, ‘[SEP]’
- **NorBERT3-base:** ‘[CLS]’, ‘ En’, ‘ samling’, ‘ kalla’, ‘ fakta’, ‘[SEP]’

System	Size	Acc, loose	F1, loose	Acc, strict	F1, strict	CPU
Solely, 1 epoch	262 kB	95.83	95.28	92.66	92.93	0.59
Solely, 5 epochs	262 kB	96.03	95.66	92.69	93.15	0.57
With a teacher, 1 epoch	97.1 kB	96.09	95.80	92.76	93.24	0.70
With a teacher, 4 epochs	97.1 kB	96.29	95.91	93.02	93.40	0.55

Table 6: Knowledge distillation. Accuracy and F1-score are given in percents.

Even though this DistilBERT tokenization looks like a valid one when compared to the previous one, again it resulted in tokens that may be present in more than one Scandinavian language. We suppose that the reason why it is so difficult to outperform static word embeddings by BERTs for this task is that all BERTs are trained on subword tokens which is a heavy disadvantage when sentences in two languages may be different by e.g. verb endings like it happens with Bokmål and Nynorsk.

However, FastText models are not robust towards out of vocabulary words. For example, there was a Bokmål single-word sentence ‘*Venner.*’ (‘Friends.’) erroneously classified by GlotLID as ‘other’. If we try to tokenize it with GlotLID, we will see that both the word and the dot are the same token and it is not present in the vocabulary. We believe this is the reason why it was difficult to get a result higher than the original GlotLID when doing knowledge distillation with a student model based on its features: the proportion of OOV tokens was the same in both cases. The simplest solution for this problem may be training a FastText model on a large Scandinavian corpus (e.g. Wikipedia, see ‘Limitations’ section) with NorBERT3-base tokenizer and use it as the source of features.

6 Conclusion

Our experiments have shown that, although there exist already fast and reliable pretrained models for language identification, their performance may be improved by fine-tuning on a particular set of languages or on texts from a specific domain.

We have also shown that BERT-like models outperform FastText methods in accuracy and F1 metrics (but not in prediction speed, which still makes them useless for many real-world tasks) on a selected test set and may be successfully used for making FastText-based models faster while preserving their accuracy. The BERT model of a ‘small’ size can be enough. Choice of the model depends on the subset of target languages: a model pretrained on one language only may perform better

than a multilingual one if all target languages are similar to each other (probably, because of a more reasonable vocabulary).

Our main contribution is a model of satisfactory quality that runs in less than a second on approximately 7000 samples.

Future work beyond directions already meant in the previous sections and in the Limitations section should also include testing the models on other datasets than a short news one and more in-depth error analysis. It also makes sense to explore other methods of BERT inference optimization: speaking honestly, the ONNX default methods used in our study are only baseline ones.

Ethical impact

We do not expect any significant ethical impact from this study. However, we did not check our training corpus for any inappropriate content. BERT-like models were reported for making predictions biased towards particular population groups (Ahn and Oh, 2021), (Jentsch and Turan, 2022) but exploring both the base models and our fine-tuned ones for biases was beyond the scope of our study.

Limitations

We experimented with enriching the training dataset with Wikipedia abstracts data (with Icelandic serving for enriching of the class ‘other’, assuming that it could play a role of ‘negative samples’). We got lower results on the development dataset for all models we trained. This makes us suspicious that the news test dataset is somewhat specific and that our models would perform bad on texts from other domains.

We also tried enriching the training dataset with a ‘silver’ dataset, which was the original training dataset translated into our other target languages. It did not include the ‘other’ label. The results also dropped, although less than with Wikipedia data. Our hypothesis is that our model learns semantics more, than surface features and the texts

on the same topics are recognized to be the same for it, but an error analysis is required to prove this assumption.

We tried to fine-tune the NorBERT3-large¹⁵ model with different learning rates and on different subsets (also on the Wikipedia one) and got extremely low metrics like F1 0.1 on the development set. Although the reasons could be pure technical e.g. if we used a wrong tokenizer when preprocessing the data, the problem is worth of further exploration.

References

- Jaimeen Ahn and Alice Oh. 2021. [Mitigating language-dependent ethnic bias in BERT](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 533–549, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. [Enriching word vectors with subword information](#). *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Laurie Burchell, Alexandra Birch, Nikolay Bogoychev, and Kenneth Heafield. 2023. [An open dataset and model for language identification](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 865–879, Toronto, Canada. Association for Computational Linguistics.
- 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](#). *CoRR*, abs/1911.02116.
- 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.
- Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.
- Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. [Distilling the knowledge in a neural network](#). *ArXiv*, abs/1503.02531.
- Sophie Jentzsch and Cigdem Turan. 2022. [Gender bias in BERT - measuring and analysing biases through sentiment rating in a realistic downstream classification task](#). In *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*, pages 184–199, Seattle, Washington. Association for Computational Linguistics.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. [Bag of tricks for efficient text classification](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.
- Amir Hossein Kargaran, Ayyoob Imani, François Yvon, and Hinrich Schütze. 2023. [GlotLID: Language identification for low-resource languages](#). In *The 2023 Conference on Empirical Methods in Natural Language Processing*.
- Joakim Nivre, Daniel Zeman, Filip Ginter, and Francis Tyers. 2017. [Universal Dependencies](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, Valencia, Spain. Association for Computational Linguistics.
- David Samuel and Lucas Charpentier. 2024. [A journey from tokenization to instruction-finetuning \(normis-tral\)](#). LTG research seminar.
- David Samuel, Andrey Kutuzov, Samia Touileb, Erik Velldal, Lilja Øvrelid, Egil Rønningstad, Elina Sigdel, and Anna Palatkina. 2023. [NorBench – a benchmark for Norwegian language models](#). In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 618–633, Tórshavn, Faroe Islands. University of Tartu Library.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC² Workshop*.
- Vésteinn Snæbjarnarson, Annika Simonsen, Goran Glavaš, and Ivan Vulić. 2023. [Transfer to a low-resource language via close relatives: The case study on Faroese](#). In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 728–737, Tórshavn, Faroe Islands. University of Tartu Library.
- 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*, volume 30. Curran Associates, Inc.

A Training details

We ran all our training experiments on NVIDIA A100 80GB. We used the learning rate 5e-5 for all transformer models, except for NorBERT 3 xs, for

¹⁵<https://huggingface.co/lgt/norbert3-large>

which we used the learning rate $5e-3$. We used the learning rate $1e-3$ for the student model.

2 Pretraining Tiny Language Models track

An empirical study of recent optimizers in tiny language models

Sander Finnset Ørnes
University of Oslo
sanderfo@math.uio.no

Abstract

The purpose of this paper is to assess how a decoder based tiny language model performs with various optimizers. The jungle of optimizers is vast, and we limit ourselves to six different optimizers. The AdamW optimizer will serve as a baseline, and the remaining five optimizers will be measured against AdamW. These other five optimizers are LAMB, Lion, Sophia, and schedule free versions of SGD and AdamW, they are all quite new, and empirical studies on these optimizers are therefore limited. Since different optimizers perform best with different learning rates we must tune the learning rate, this is done through evaluation on the LAMBADA dataset and on the BLiMP dataset. Furthermore, we test some different schedulers and weight decays. Finally, a brief qualitative analysis will be given for some predictions made by the best model on the LAMBADA dataset.

1 Introduction

Language models are getting bigger and bigger, enabled by the ever increasing computational capacity of computers. The most famous family of language models, the Generative Pre-Trained Transformer (GPT) family, for instance, have only increased in size, from the "modest" size of 117 million parameters in the GPT-1 model (Radford et al., 2019), to 175 billion parameters used by the GPT-3 model (Brown et al., 2020). The GPT-4 model has an unknown size, but the number of parameters is rumoured to be in the trillions. It is certainly possible that language models need to be of such a magnitude to be comparable to humans in terms of language understanding, but it is also possible that a few of those trillions of parameters are superfluous. It would be impossible to test for the significance of all these parameters, but to test the hypothesis that some of the parameters of these massive language models are redundant, we could instead show that smaller models have an equally

good performance. This will likely require a lot of experimentation, and one component of language models that directly affects the parameters is the choice of optimization method. In addition to creating models with more efficient parameters, another motivation is to speed up convergence, and thereby reduce the training cost.

Optimizers control the update of the parameters for each step in a training process by minimizing a loss function, the exact way in which this is done varies greatly and is an interesting topic in and of itself. However, mathematical and implementational details on optimizers are way beyond the scope of this paper, we will therefore limit ourselves to some basic properties and refer elsewhere for details.

In our experiment we have tried several optimizers, and for each of them, several models have been trained with varying learning rates. The learning rate is perhaps the most important hyperparameter of an optimizer, and it controls the speed at which optimizers update the parameters based on the loss. Too large of a learning rate might overshoot the global minimum, and too small of a learning rate might get stuck in a local minimum, or converge really slowly. Different optimizers work best with different learning rates, therefore we perform a grid search over the chosen optimizers and some learning rates.

Evaluation of the models will be done on both the BLiMP dataset (Warstadt et al., 2023b) and the LAMBADA dataset (Paperno et al., 2016). First we determine the best learning rate for each optimizer, then, we will test some different schedulers and weight decays given these best learning rates. Note that this is not really best practice from a statistical point of view, but is done to reduce the amount of possible combinations. After the exploration of the hyperparameter space, some predictions made by the best model according to the LAMBADA dataset will be analyzed qualitatively.

2 Related works

As a consequence of the the recency of the optimizers we consider, there are few empirical studies comparing them. The only one we have found is [Kaddour et al. \(2023\)](#), which uses AdamW as a baseline, and compares it with the Sophia optimizer and the Lion optimizer by testing them on the GLUE dataset ([Wang et al., 2019](#)) and the SuperGLUE dataset ([Wang et al., 2020](#)) for both a BERT-type model and a T5-type model. They also develop an evaluation methodology called *reference system time*, as they claim that "evaluation methodology is not standardized" ([Kaddour et al., 2023](#), p. 1).

Other, and earlier comparisons exist. The most comprehensive one is perhaps ([Schmidt et al., 2021](#)), which compares the most popular optimizers at the time on a variety of problems, datasets and model designs. They conclude that, of the optimizers considered, no optimizer is universally better, but that Adam is often a good choice.

In ([Wilson et al., 2018](#)) the angle is adaptive optimizers, such as Adam, vs non-adaptive optimizers, such as SGD. They consider a few different deep learning experiments, but they also construct a very interesting example where the adaptive methods fail to make the correct predictions in a linearly separable binary classification task and make the following bold claim in their conclusion: "our experimental evidence demonstrates that adaptive methods are not advantageous for machine learning", but bear in mind that the initial year of publication was 2017.

Other, and similar works include [Choi et al. \(2020\)](#) and [Schneider et al. \(2019\)](#), both consider SGD, momentum and Adam. The former compares a few more adaptive optimizers, and concludes that "RMSPROP, ADAM, and NADAM never underperformed SGD, NESTEROV, or MOMENTUM under our most exhaustive tuning protocol" ([Choi et al., 2020](#), p. 8), i.e. the adaptive methods they tested did not underperform the non-adaptive methods they tested. The latter also develops a python package of deep learning optimization benchmarks called "DeepOBS", and concludes similarly to [Schmidt et al. \(2021\)](#), that none of the tried optimizers are best for every problem, but that Adam "compares favorably on most test problems" ([Schneider et al., 2019](#), p. 9).

3 Dataset

As discussed, models can be very large in terms of parameters, they can also be trained on huge amounts of data, far exceeding the amount of words a human is exposed to throughout their life. As with parameter size, the largest models are trained on datasets containing trillions of words, the training dataset we will use is much smaller and is similar to the small dataset from the BabyLM challenge ([Warstadt et al., 2023a](#)). It contains 10 million words from various English sources and is supposed to mimic the words a thirteen-year-old child could have been exposed to throughout their lives, see table 1.

4 Model

The model is decoder based with self attention. It uses the GEGLU activation function ([Shazeer, 2020](#)) on each layer, and the GELU activation function on the classification head. The feed forward neural network part of the model has three hidden layers each of a size of 192. There are three attention heads, and the vocabulary size is set to 3072. Additionally, all dropout probabilities are set to 0.1. Furthermore, each layer has layer normalization ([Ba et al., 2016](#)). Note that 10000 steps with a batch size of 256 corresponds to 15 epochs on our dataset. The total amount of parameters amounts to 5 965 056.

As mentioned, the only hyperparameters we will be tuning in the proper way are the optimizer, and the peak learning rate of the optimizer. But we will also test some different learning rate schedulers, and some different weight decays, given the best combinations of optimizer and learning rate. For our learning rate experiments, we use a linear scheduler and a weight decay of 0.1. The warmup will be linear and last for 1.6% of the total amount of steps. The final learning rate will, for all runs with linear and cosine scheduling, be 10% of the peak learning rate. In addition, whenever relevant we set the optimizer parameters $\epsilon = 1e - 6$, and $\beta = (0.9, 0.98)$. Lastly, the seed is set to 42 for all runs. Altering the parameters ϵ , β and seed value might single-handedly, or in collaboration, change the results of our forthcoming experiment. This highlights a weakness of our study, as we do not know how sensitive the results are to changes in these parameters.

Source	Domain	Words	Proportion
British national corpus (BNC)	Dialogue	0.8M	8%
Childes (Macwhinney, 2000)	Child-directed speech	2.9M	29%
Project Gutenberg (Gerlach and Font-Clos, 2018)	Written fiction and nonfiction	2.6M	26%
OpenSubtitles (Lison and Tiedemann, 2016)	Scripted dialogue	2M	20%
Simple English Wikipedia	Written fiction	1.5M	15%
Switchboard (Stolcke et al., 2000)	Written fiction	0.1M	1%

Table 1: Dataset

5 Optimizers

The collection of optimizers that are commonly used in deep learning algorithms has itself come into existence through a sequential optimization procedure, where authors have built upon previous work and tried to mitigate the weaknesses of previously proposed optimizers. Perhaps the most basic optimizer is (deterministic) gradient descent, which finds the closest local minimum. If a problem has many local minimums, the probability that gradient descent will find the global minimum is slim. The method of stochastic gradient descent, on the other hand, alleviates this issue and is able to escape local minima, which could lead to better models. On the backbone of stochastic gradient descent, a myriad of other optimizers have come to life. Such as Root mean square propagation (RMSProp) and Adaptive Gradient algorithm (Adagrad). We refer to Ruder (2017) for a quick overview of these and some other optimization algorithms. The focal point of our study will be on newer optimizers, the most popular type of optimizers, namely Adam type optimizers, will set the bar for these newer methods.

5.1 AdamW

A very popular optimization algorithm is the Adaptive Moment Estimation algorithm (Adam) (Kingma and Ba, 2017), which essentially allows for individual learning rates for each parameter, set dynamically at each iteration, by using estimates of the first and second moments of the gradient. This is fundamentally different from standard stochastic gradient descent (SGD) which uses the same learning rate for each parameter. As a consequence of this adaptivity, using the Adam optimizer often leads to faster convergence than non-adaptive methods such as SGD. Further enhancements to the Adam algorithm have been suggested, resulting in optimizers such as Rectified Adaptive Moment Esti-

mation (RAdam) (Liu et al., 2021), Nesterov Adaptive Moment Estimation (NAdam) (Dozat, 2016), and AdamW (Loshchilov and Hutter, 2019). At the point of writing, AdamW is maybe the most popular optimizer on the market, it attempts to improve upon the standard Adam algorithm by decoupling the weight decay and learning rate, hence reducing their dependence of one another. Because of the sequential relationship between the optimizers mentioned above, we will only include what might be the most popular one in our experiments, namely the AdamW optimizer.

5.2 LAMB

In addition to some of the aforementioned optimization methods, we will also experiment with the LAMB algorithm (You et al., 2020), this is a further development of the AdamW algorithm which adds layerwise adaptivity to the learning rate in addition to the parameter adaptivity of the basic Adam algorithm, this means that the learning rate is individual to the parameter and the layer, and not just the parameter. The main motivation behind the LAMB algorithm is to speed up the training process by allowing for large batch learning without degrading performance, much larger than batch sizes of 256 which we will be using. See (You et al., 2020) for details.

5.3 Lion

There is an even newer optimization method called Lion (EvoLved Sign Momentum) (Chen et al., 2023), which is also motivated by the wish to effectivize the training process, both in terms of time and memory. The memory reduction compared to the Adam type algorithms is achieved by only tracking the first moment of the gradient, compared to both first and second moment for the Adam algorithms. This also makes the Lion optimizer simpler to implement. We have not discussed the hyperparameters of the optimizers thus far, but this is an

appropriate time to begin. Because of the mechanics of the Lion algorithm, the authors suggest that the learning rate should be 3-10x smaller than the learning rate for AdamW, and further commands that "initial value, peak value, and end value of the learning rate should be changed *simultaneously* with the same ratio compared to AdamW" (Chen et al., 2023, p. 14). This criteria will be satisfied in our case since we compare models trained with a linear warmup using the same amount of warmup steps and the same scheduler, which means that the start learning rates, peak learning rates, and end learning rates will have the same ratio. In addition, they suggest increasing the weight decay by 3-10x as compared to AdamW. Our main focus is on learning rate, but we will also try some different values of weight decay, as suggested by the authors.

5.4 Sophia

All the above methods are first-order methods, meaning that they only require computation of the gradient, and not the second-order structure, namely the Hessian. The reason for this is that computation of the Hessian is very costly. Instead of computing the actual Hessian, the **Second-order Clipped Stochastic Optimization** (Sophia) algorithm (Liu et al., 2024) instead approximates the diagonal of the Hessian on every n 'th step, in our case chosen to be 10. This has a minimal computational and per-step time overhead according to the authors. In addition, the authors claim a 2x speed up in terms of "number of steps, total compute, and wall-clock time" (Liu et al., 2024, p. 9) as compared to AdamW for the models they consider, which are GPT-2 models of various sizes. For hyperparameters they recommend the learning rate to be 0.8 of the learning rate used for AdamW, and 2x the weight decay.

5.5 Schedule-free optimizers

Finally, we will also try two schedule-free optimization methods, these are brand new, one is a schedule-free version of SGD and the other is a schedule-free version of AdamW. "Schedule-free" means that it is not necessary to use a learning rate scheduler. See (Defazio et al., 2024) for details on the algorithms. In this case they suggest a learning rate 10-50x larger for the schedule free SGD as compared to the standard SGD, and 1-10x larger for the schedule free AdamW as compared to the standard AdamW. The authors also hints that other parameters, such as weight decay, might need

tuning to be comparable to the scheduled versions.

6 Evaluation

As mentioned above, evaluation will first be done on the optimizers and learning rates in a grid with the largest peak learning rate being 0.05, and the smallest being 0.0001. This is done to cover the range of reasonable learning rates for a problem of this type, and also to take into account the suggestions for the different optimizers made by the respective authors. Then we try some different schedulers and weight decays for the best combinations of optimizer and learning rate found. All evaluations will be done on the BLiMP dataset and on the LAMBADA dataset. In addition we use a random untrained decoder as a baseline. Finally, the best scoring model, according to LAMBADA accuracy will be used to qualitatively analyze some examples in the LAMBADA dataset.

6.1 BLiMP

The Benchmark of Linguistic Minimal Pairs (BLiMP) is a dataset used to test models on various grammatical problems (Warstadt et al., 2023b). The dataset consists of grammatically correct and grammatically incorrect sentences, where the job of the model is to predict which sentences are grammatically correct, and which are not. Quoting the authors: "BLiMP consists of 67 minimal pair paradigms, each with 1,000 sentence pairs in mainstream American English grouped into 12 categories." (Warstadt et al., 2023b, p. 379), Where each category represents a grammatical phenomena. It is possible to do a very fine-grained evaluation of a model using these 67 minimal pair paradigms, since each of them has its own score, and thereby diagnosing specific strengths and weaknesses of the model in question. To simplify the evaluation we only look at the average score over these 67 minimal pair paradigms. Note that guessing randomly on the BLiMP dataset gives an expected score of 50%, therefore, none of our models should score significantly less than 50%. The random decoder illustrates this fact with a score of 49.29%.

6.2 LAMBADA

In addition to BLiMP scores, we will also use the LAMBADA (LAnguage Modeling Broadened to Account for Discourse Aspects) (Paperno et al., 2016) dataset to evaluate the models. The task is to predict missing words in sentences, where the correct word should be fairly easy to guess given

the full context, but not easy at all given no context beyond the sentence of the missing word. This is usually quite easy for humans, but, as we shall soon see, difficult for language models. Substantially, the LAMBADA dataset tests the memory of a language model. We focus on the accuracy scores on the LAMBADA dataset, but perplexity scores will also be mentioned when deemed fitting. As usual, the larger the accuracy the better, and the opposite for perplexity scores, that is, lower perplexity scores are better. The random decoder gets an accuracy score of 0% on the LAMBADA dataset.

7 Results

7.1 Optimizer and learning rate grid search

We now compare the various optimizers and peak learning rates based on LAMBADA accuracy and average BLiMP accuracy. Firstly, we can see that the LAMBADA scores and BLiMP scores are consistent with each other in the sense that they assign the best score to the same learning rates for each respective optimizer. The very best optimizer and learning rate combination is Adamfree with a learning rate of 0.01 for LAMBADA, and AdamW with a learning rate of 0.005 for BLiMP.

Studying the LAMBADA accuracies from table 2 we can see many 0's. In particular, the SGDfree optimizer never gets a positive accuracy score. However, the perplexity scores on the LAMBADA dataset is the smallest for a learning rate of 0.5 and the largest for a learning rate of 0.0001, but all very poor. It also performs consistently poor on the BLiMP dataset, and only marginally better than guessing which would give a score of about 50%. This is probably a result of the amount of steps taken, with more steps, SGDfree should eventually be comparable to the other optimizers, which are all fast-converging optimizers. But at the very least, the perplexity scores indicate that a larger learning rate for the SGDfree is appropriate, as claimed by the authors.

All the other optimizers have a LAMBADA accuracy above 8% and below 9% for their best learning rate, and all achieve a BLiMP score better than 69.5%. AdamW is, in this case, the most robust optimizer in terms of learning rates with accuracy scores exceeding 6.5% for learning rates between 0.01 and 0.001. On the BLiMP dataset, the same trend occurs, where AdamW scores above 67.5% for all learning rates between 0.01 and 0.0005.

Comparing the best learning rates for each optimizer, we can clearly see that the suggestions for learning rates made by the respective authors match quite well, even without any tuning of weight decay.

For Adamfree, the authors suggested a learning of 1-10x bigger than that for AdamW, which fits quite well with our results since AdamW performed best with a learning rate of 0.005, and Adamfree performed best with a learning rate of $2 \cdot 0.005 = 0.01$, and also quite good with a learning rate of 0.005.

The Lion optimizer was suggested to perform best with learning rates 3-10x smaller than that for AdamW, the best learning rate for the Lion optimizer was $0.005/10 = 0.0005$, which just fits in their suggested range, however, based on the good score AdamW achieved with a learning rate of 0.0025, it is possible that the ideal learning rate for AdamW is smaller than 0.005. It is also not unlikely that the optimal learning rate for the Lion optimizer is slightly larger than 0.0005, so that their recommendations are right on the mark

For the LAMB optimizer the authors did not make a specific suggestion, but looking at table 4 in You et al. (2020, p. 8), we can see that they have used a learning rate of $\frac{5}{20.5 \cdot 10^3}$ for BERT training with a batch size of 16K, and, that the best learning rate with a batch size of 16K for the AdamW optimizer was 0.0002, which is smaller, see table 8 in You et al. (2020, p. 19). This is, of course, a different setting, but it is perhaps sensible to use a slightly larger learning rate for the LAMB optimizer compared to AdamW since LAMB uses a per parameter, *and per layer* normalization. In our experiment at least, the LAMB optimizer performed the best with a learning rate of $2 \cdot 0.005 = 0.1$, and similarly to AdamW and Adamfree, it is also quite robust to various learning rates.

Finally, the Sophia optimizer performed best for the learning rate 0.001, which is smaller than the best learning rate for AdamW, and larger than the best learning rate for Lion, this is in line with the authors suggestions, however, their specific recommendation was 0.8 of the learning rate for AdamW, or 3-5x that of the Lion optimizer. This would have given a larger optimal learning rate for Sophia than the best we have found, by looking at the scores Sophia achieved with a learning rate of 0.0005, it seems like the very precise suggestions made by the authors are not completely congruous to our setting. But note that the recommendation in Liu et al.

(2024) implies a narrower learning rate interval for the Lion optimizer compared to AdamW, than the recommendation made by the inventors of Lion in Chen et al. (2023), where they say that the learning rate of the Lion should be 3-10x smaller than that of AdamW.

Summing up, there are few surprises in table 2, and table 3, and for the most part our results are in line with what the different authors suggested in regards to learning rate, with Lion being the only exception. Now we move over to the "illegal" part of our experiments, where, for the best learning rates for each optimizer, different schedulers, and different weight decays will be tried. As previously stated, this should ideally be done as a grid search in combination with the learning rates, but to simplify matters, we perform these experiments only for the best optimizer and learning rate combinations. The issue with this procedure is, of course, that the same dataset is used to tune the learning rate first, and then also used to evaluate different schedulers and weight decays. This will bias our optimizers and learning rate combinations in favor of a linear scheduler with weight decay set to 0.1, since these were the parameters used in the grid search experiment.

As a final remark, we note that the training time was around 34-35 minutes for most runs with the optimizers AdamW, Lion, Adamfree, and SGDfree. For the LAMB optimizer, most runs took around 37-38 minutes, and for the Sophia optimizer, it was around 38-39 minutes.

7.2 Experimentation with schedulers

The results in this section are not very surprising either. The linear scheduler, and the cosine scheduler were comparable in performance, then the inverse square root was the third best, and running with no scheduler performed the worst. In figure 1 we can see how the different schedulers affect the learning rate over a total run. The figure also shows why the cosine and linear scheduler performs similarly, it is simply because the learning rate is quite similar at all steps. The linear scheduler slightly edges out the cosine scheduler on the BLiMP dataset, but this is not unnatural given that the learning rates were tuned with linear schedulers. The results with the inverse square root scheduler is quite close, and using a slightly larger peak learning rate for the inverse square root optimizer as compared to linear and cosine could give similar results. Running with no scheduler, on the other hand, likely requires

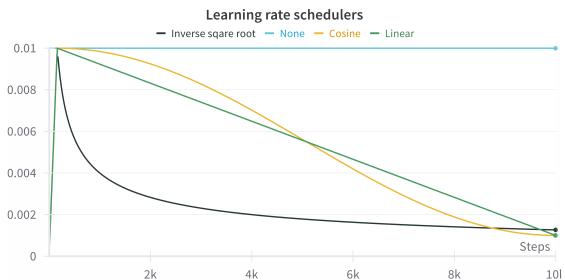


Figure 1: Visualization of different schedulers

a smaller learning rate as compared to linear and cosine scheduler.

From tables 4 and 5 we can see that only the Lion optimizer with a learning rate of 0.0005 diverges from the ranking pattern described above, it performs better without a scheduler than with the inverse square root scheduler on both BLiMP and LAMBADA, and also performs better on BLiMP than a cosine scheduler. This might be because the learning rate used for the experiments with the Lion optimizers was 0.0005, i.e. quite small, and therefore that the need for scheduling is reduced, this also suggests that the learning rate of 0.0005 is slightly smaller than the ideal learning rate for the Lion optimizer, as per reference to our discussion above. All in all the differences are rather small, especially on the BLiMP dataset. These experiments naturally exclude the schedule-free optimizers.

7.3 Experimentation with weight decay

Above we saw that both a linear scheduler and a cosine scheduler worked well in our setting. This section will instead experiment with the weight decay parameter, similarly to the experiments for scheduler, this will be done with the best learning rate for each optimizer found in the grid search. Recall that this grid search was performed with the baseline value of 0.1 for weight decay, and with a linear scheduler for all non-schedule free optimizers. Based on the poor results achieved by the SGDfree optimizer above, we skip experimentation with the SGDfree optimizer.

The results can be found in table 6 and 7. Unlike the previous experiment with schedulers, there is less of a trend in this case, but perhaps a very slight trend indicating that the larger weight decays were a little bit better, given the bias towards a weight decay of 0.1, a grid search with weight decay and learning rate might accentuate this possible trend further. Again, the differences, on the

Learning rate	AdamW	LAMB	Lion	Sophia	Adamfree	SGDfree
0.05	0.00	0.00	0.00	0.00	0.00	0.00
0.01	6.91	8.48	0.00	0.00	8.97	0.00
0.005	8.75	7.14	0.00	0.00	7.84	0.00
0.0025	8.13	6.50	0.00	0.00	5.39	0.00
0.001	6.66	2.15	4.81	8.19	4.72	0.00
0.0005	4.17	0.04	8.13	7.14	1.40	0.00
0.0001	0.00	0.00	5.20	1.96	0.00	0.00

Table 2: LAMBADA accuracy (in percent). The bold numbers indicates best score in the column, and the italic text indicates that the value is also the best in the entire table.

Learning rate	AdamW	LAMB	Lion	Sophia	Adamfree	SGDfree
0.05	58.62	58.27	52.95	53.12	59.29	54.97
0.01	70.41	69.97	56.13	59.41	69.65	54.54
0.005	70.53	68.31	57.85	60.24	69.48	53.42
0.0025	70.19	68.15	59.26	62.53	69.54	53.06
0.001	68.56	64.75	68.39	70.20	68.27	53.30
0.0005	67.53	61.37	69.68	69.59	65.44	52.88
0.0001	62.88	56.66	67.28	65.82	60.92	50.70

Table 3: Average BLiMP accuracy (in percent)

Scheduler	AdamW (0.005)	LAMB (0.01)	Lion (0.0005)	Sophia (0.001)
Linear	8.75	8.48	8.13	8.19
Cosine	8.95	8.42	7.96	8.50
Inverse square root	7.53	7.72	5.45	6.57
None	5.08	6.25	6.81	5.82

Table 4: LAMBADA accuracy (in percent) for the non schedule-free optimizers for 4 different schedulers at their best learning rate. The best learning rate is in parenthesis.

Scheduler	AdamW (0.005)	LAMB (0.01)	Lion (0.0005)	Sophia (0.001)
Linear	70.53	69.97	69.68	70.20
Cosine	69.72	69.18	68.93	70.11
Inverse square root	68.91	68.97	68.75	69.37
None	66.60	68.15	69.22	67.35

Table 5: Average BLiMP accuracy (in percent) for the non schedule-free optimizers for 4 different schedulers at their best learning rate

BLiMP dataset in particular, are small. As with the schedulers experiment, the LAMB optimizer seems to be the most robust to variations, and it performs well for all the tested weight decays, this could be a result of the double normalization procedure that it performs. The worst weight decay on the BLiMP dataset for the LAMB optimizer was actually 0.1, which is interesting since this weight decay was used for the learning rate tuning.

On the other hand, the Lion optimizer seems to be the most sensitive to changes. The winner of these experiments are AdamW, which has the best score on the LAMBADA dataset with a learning rate of 0.2, and the best score on the BLiMP dataset with a learning rate of 0.4.

7.4 Examples of LAMBADA predictions

To round off this section we briefly, and informally, analyze a couple of examples produced by the model scoring 9.7 on the LAMBADA dataset, that is, the model trained with AdamW optimizer with a peak learning rate of 0.005, weight decay equal to 0.2, and a linear scheduler. The task is to predict the missing word, highlighted by and underscore.

Example 1: "Only one source I know of that would be likely to cough up enough money to finance a phony sleep research facility and pay people big bucks to solve crimes in their dreams," Farrell concluded dryly. "What can I say?" Ellis unfolded his arms and widened his hands. "Your tax dollars at work." Before Farrell could respond, Leila's voice rose from inside the house. "No insurance?" she wailed. "What do you mean you don't have any _?". The correct answer is "insurance", but the model outputs "moneyurance", which is not correct, but has the common part "urance" at the end. Also the first part of the word "money" is semantically related to "insurance", even though the words themselves are not similar. It almost seems as if the model can not decide if it prefers "money" or "insurance", and instead collapses them into one word.

Example 2: "He heard Rhinna speak "The Queen wants you in her carriage." Tom spoke "No, I'm not going in some asylum." Ran was seen standing next to him spoke "It's just for a private talk with you that's all." Tom groaned and went inside the carriage to sit down next to the _". The answer is "Queen", and the prediction is "carueen". In this example the prediction is not an actual word, but still similar to the correct word "Queen", the prediction is also phonologically quite similar, but

this could be a coincidence as the model likely combine the word "Queen" with "carriage".

There were other similar examples. For example "thelen" was predicted when the correct word was "Helen", and "doastin" was predicted, when the correct answer was "Hastin".

8 Conclusion

We have examined a part, and hopefully the most important part, of the hyperparameter space for various optimizers in the context of a decoder based tiny language model with less than 6 million parameters. All models were evaluated on both the BLiMP dataset and the LAMBADA dataset, and we have seen that all tested optimizers, except SGDfree, perform similarly for their respective best learning rates. We have also seen that varying the scheduler and the weight decay could improve the models. Concluding that all 5 fast-converging optimizers are approximately equally good given proper tuning, is quite a boring result, but it aligns with the results of the papers mentioned in section 2 on related works. Perhaps the most thorough of them was the paper [Schmidt et al. \(2021\)](#). It tests several different optimizers, amongst them Adam, on several different tasks. Their set of tested optimizers does not overlap our set of tested optimizers, but it is quite likely that applying the Adam algorithm to our problem would have led to a performance similar to that of AdamW (with appropriate tuning). Their conclusion was that no optimizer is universally better than all other optimizers, but nonetheless that the Adam optimizer is usually amongst the best performing optimizers across a variety of problems. They also say that "Evaluating multiple optimizers with default hyperparameters works approximately as well as tuning the hyperparameters for a fixed optimizer." ([Schmidt et al., 2021](#), p. 2). Our conclusion is similar, with proper tuning, all optimizers except SGDfree performs similarly. The one that comes out on top is the AdamW optimizer, but perhaps more important than the marginally better scores that it achieved compared to the other optimizers, is that AdamW performs well for quite a large range of learning rates, making it easier to tune than competitors like Lion and Sophia. LAMB and Adamfree also shares this property of good performance across a large range of parameters. Additionally, Adamfree has the benefit of freeing us of the need to tune the scheduler, consequently it might become a popular

Weight decay	AdamW (0.005)	LAMB (0.01)	Lion (0.0005)	Sophia (0.001)	Adamfree (0.01)
0.01	7.14	8.00	6.87	6.50	7.08
0.05	8.3	9.06	6.89	7.01	7.39
0.1	8.75	8.48	8.13	8.19	8.97
0.2	9.7	9.02	6.79	8.19	7.90
0.3	8.87	8.87	6.93	7.67	7.20
0.4	8.42	9.1	6.95	8.42	8.15

Table 6: LAMBADA accuracy (in percent) for 6 different weight decays

Weight decay	AdamW (0.005)	LAMB (0.01)	Lion (0.0005)	Sophia (0.001)	Adamfree (0.01)
0.01	69.16	70.18	68.81	69.57	70.08
0.05	70.15	70.16	68.97	70.52	69.77
0.1	70.53	69.97	69.68	70.20	69.65
0.2	70.55	70.07	69.84	70.20	69.42
0.3	69.77	70.28	69.79	70.25	69.90
0.4	71.03	70.16	69.87	70.67	68.20

Table 7: Average BLiMP accuracy (in percent) for 6 different weight decays

optimizer in the near future.

9 Future work

In future work, it would be interesting to robustify our results. Firstly, one could try different seeds and see how much the results change, since this is perhaps the biggest weakness of our study. It would also be interesting to tune the β and ϵ values that several of the optimizers use. These were fixed and equal for all optimizers in our experiment, but might have a large impact on the performance of the various optimizers. It is also possible that altering these values would change our conclusion regarding AdamW, Adamfree and LAMB being easier to tune than Lion and Sophia. There are several other natural extensions, for instance, altering the model architecture or testing on other datasets.

References

- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. [Layer normalization](#). *Preprint*, arXiv:1607.06450.
- 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](#). *Preprint*, arXiv:2005.14165.
- Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. 2023. [Symbolic discovery of optimization algorithms](#). *Preprint*, arXiv:2302.06675.
- Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, and George E. Dahl. 2020. [On empirical comparisons of optimizers for deep learning](#). *Preprint*, arXiv:1910.05446.
- Aaron Defazio, Xingyu (Alice) Yang, Konstantin Mishchenko, Ashok Cutkosky, Harsh Mehta, and Ahmed Khaled. 2024. [Schedule-free learning - a new way to train](#).
- Timothy Dozat. 2016. Incorporating Nesterov Momentum into Adam. In *Proceedings of the 4th International Conference on Learning Representations*, pages 1–4.
- Martin Gerlach and Francesc Font-Clos. 2018. [A standardized project gutenber corpus for statistical analysis of natural language and quantitative linguistics](#). *Preprint*, arXiv:1812.08092.
- Jean Kaddour, Oscar Key, Piotr Nawrot, Pasquale Minervini, and Matt J. Kusner. 2023. [No train no gain: Revisiting efficient training algorithms for transformer-based language models](#). *Preprint*, arXiv:2307.06440.
- Diederik P. Kingma and Jimmy Ba. 2017. [Adam: A method for stochastic optimization](#). *Preprint*, arXiv:1412.6980.
- 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).
- Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. 2024. [Sophia: A scalable stochastic second-order optimizer for language model pre-training](#). *Preprint*, arXiv:2305.14342.
- Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2021. [On the variance of the adaptive learning rate and beyond](#). *Preprint*, arXiv:1908.03265.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). *Preprint*, arXiv:1711.05101.
- Brian Macwhinney. 2000. [The chldes project: tools for analyzing talk](#). *Child Language Teaching and Therapy*, 8.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. [The lambda dataset: Word prediction requiring a broad discourse context](#). *Preprint*, arXiv:1606.06031.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language models are unsupervised multitask learners](#).
- Sebastian Ruder. 2017. [An overview of gradient descent optimization algorithms](#). *Preprint*, arXiv:1609.04747.
- Robin M. Schmidt, Frank Schneider, and Philipp Hennig. 2021. [Descending through a crowded valley - benchmarking deep learning optimizers](#). *Preprint*, arXiv:2007.01547.
- Frank Schneider, Lukas Balles, and Philipp Hennig. 2019. [Deepobs: A deep learning optimizer benchmark suite](#). *Preprint*, arXiv:1903.05499.
- Noam Shazeer. 2020. [Glu variants improve transformer](#). *Preprint*, arXiv:2002.05202.
- Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. [Dialogue act modeling for automatic tagging and recognition of conversational speech](#). *Computational Linguistics*, 26(3):339–373.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2020. [Superglue: A stickier benchmark for general-purpose language understanding systems](#). *Preprint*, arXiv:1905.00537.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. [Glue: A multi-task benchmark and analysis platform for natural language understanding](#). *Preprint*, arXiv:1804.07461.
- Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023a. [Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.
- Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2023b. [Blimp: The benchmark of linguistic minimal pairs for english](#). *Preprint*, arXiv:1912.00582.
- Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. 2018. [The marginal value of adaptive gradient methods in machine learning](#). *Preprint*, arXiv:1705.08292.
- Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. [Large batch optimization for deep learning: Training bert in 76 minutes](#). *Preprint*, arXiv:1904.00962.

From bad babies to good language students? - An experiment of training tiny language models on ESL data

Erling Midtgard

erlingmi@ifi.uio.no

Abstract

This paper takes inspiration from the BabyLM competition in which the goal was to pre-train tiny language models on a dataset similar to language children might be exposed to. Instead of using child-directed data, I attempted to use English as second language (ESL) data to pre-train an encoder model, a decoder-model and an encoder-decoder model. More specifically, I used the child-directed CHILDES corpus to train the baseline models and the Teacher student chatroom corpus (TSCC) for the ESL models. I used backtranslation via ten different languages to augment the TSCC data. All models were evaluated on the BLiMP and LAMBADA metrics. Overall the difference between models trained on the different datasets was very small. The best model turned out to be the CHILDES decoder, but it was only marginally better than the TSCC decoder, with average BLiMP accuracies of 70,2 % and 70,1 % respectively.

1 Introduction and background

This paper takes the BabyLM competition (Warstadt et al., 2023) as a starting point. The goal of the BabyLM competition was to explore the possibility of training small language models with a lot fewer parameters than the well-known big ones like Llama2 (Touvron et al., 2023) and BERT (Devlin et al., 2018). The training data provided for the BabyLM was a combination of different corpora that was meant to be comparable to the number of words a 13 year-old child can be expected to have been exposed to. Warstadt et al. (2023, pp.4) estimates this number to be 24 to 84 million words. The competition included three different tracks: the *strict-small* track, where the participants were provided with a dataset of 10 million tokens and only allowed to train their models on these, the *strict* where they were provided with a dataset of 100 million tokens, and a *loose* track where they in addition to the dataset of 100 million tokens, were permitted

to use non-linguistic data, for instance images and sound. The training data provided for the competition was selected to be similar to language children may meet throughout their daily lives. 56% came from transcribed speech, whereas the rest came from textual sources. 40% came from sources intended for, or deemed appropriate for children, like books or films, or transcribed child-directed speech (Warstadt et al., 2023, pp.5).

A lot of different approaches were tried for the BabyLM competition. I will give a short overview in the next section.

I find the idea of training tiny LM's on textual data with reduced linguistic complexity quite interesting. After reading some of the papers from the BabyLM competition, I had an alternative idea: Children are not the only ones that acquire a new language, adults also do when studying foreign languages. It is possible that the reason child-oriented speech is structured the way it is, i.e. has a simplified vocabulary and syntactical structure, is that children have not yet developed the cognitive capacities they need to be able to handle more advanced language. It could also be argued that it doesn't really matter if parents and kindergarten teachers use a simplified form of child-directed language, what Steven Pinker refers to as "Motherese" - children living in a society will acquire language anyway (Pinker, 1994, pp.28-29).

Adult language learners also need a small vocabulary when they start out learning a new language, but they can handle more complex concepts and syntax than children, especially if their native tongue is not too different from the new language they are studying. To use a machine learning analogy, we could say adult language learners are already "pre-trained" on their native tongue.

The way to train good tiny LM's might not be through training them on child-directed textual data, but using the simplified textual data that second language students are exposed to. The LM's

might be "bad babies"(Steuer et al., 2023), but could they be raised to good language students?

In this paper, I want to explore the possibility of pre-training tiny LM's on ESL (English as a second language) data. I will use a dataset of about 10 million tokens, similar to the BabyLM strict-small track (see table1). I will first train an encoder-model, a decoder-model and an encoder-decoder model on these data. I will then substitute the CHILDES dataset with an English as a second language (ESL) dataset. I will keep the rest of the training data and train the three aforementioned architectures on the new dataset. This will give me a total of six models, which I will evaluate using BLiMP (Warstadt et al., 2020) and LAMBADA (Paperno et al., 2016)

2 Previous work

I have not been able to find any previous works that attempt to train tiny LM's on ESL data. There are however quite a few that have done so on child-directed data. An early paper suggesting a road map for this is Dupoux (2018). Dupoux (2018) argues that there are a few puzzles to be solved when it comes to how children acquire language, namely how children are able to learn language at all given its complexity and variability across cultures. Also, it seems like children are able to acquire language spontaneously and for the most part without supervisory feedback. Dupoux (2018) calls for using deep learning models to reverse engineer children's early language development and mentions the CHILDES repository (MacWhinney, 2000) as a possible dataset for performing experiments to achieve this.

Huebner et al. (2021) trained a smaller model named BabyBERTa which was based on RoBERTa (Liu et al., 2019). Huebner et al. (2021) only used 30 million tokens for training and a model size of 5 Million parameters, compared to RoBERTa's 30 billion tokens and 125 million parameters. Huebner et al. (2021) used the AO-CHILDES corpus (Huebner and Willits, 2021) which is an age-ordered version of the CHILDES corpus (MacWhinney, 2000) as part of their training set. They also used a version of the Newsela corpus(Xu et al., 2015) which had been rewritten for children and age-ordered. Huebner et al. (2021) found that the models trained on child-directed text got a higher accuracy than those trained on text from Wikipedia.

The BabyLM challenge received 31 submissions

(Warstadt et al., 2023, pp.2). It is outside the scope of this paper to discuss them all, but I will mention a selection.

The winner of the *strict* track was Georges Gabriel Charpentier and Samuel (2023). They used a previously introduced architecture named LTG-BERT (Samuel et al., 2023) which made several changes to the transformer architecture by using additional layer normalization, the GEGLU activation function, alternative ways of calculating attention and scaled weight initialization (Georges Gabriel Charpentier and Samuel, 2023), (Warstadt et al., 2023, pp. 12). The same authors also won the *strict-small* track (Georges Gabriel Charpentier and Samuel, 2023).

The winner of the *loose* track was Xiao et al. (2023). They used a "contextualizer approach" where they mixed training data from different categories in different ways to avoid what they call the "contextualization trap" (Xiao et al., 2023, pp. 317) which is a risk of overfitting training data to the surrounding context. By combining the training data in a variety of ways and training the models for 40 epochs they received an average BLiMP accuracy of 85.54 for their best model (Xiao et al., 2023, pp. 321).

Steuer et al. (2023) used the Open Pre-trained Transformer(OPT)-architecture (Zhang et al., 2022). They trained 24 different models with varying hidden sizes. During training they used a sequence length of 64 for the first epoch, which they increased to 256 for the remaining four epochs.

Martinez et al. (2023) used a curriculum-learning approach where they increased complexity during training. They used three approaches: The first was *vocabulary curriculum* where they gradually increased the size of the vocabulary during the epochs of training. The second was *data curriculum* where they gradually increased the difficulty of the training data samples. Here, variants of perplexity were used as a measure of difficulty with increasing perplexity indicating increasing difficulty. Perplexity can be defined as

"the inverse probability of a test set, normalized by number of words(Jurafsky and Martin, pp.39)."

In other words, perplexity can be described by how "surprised" or "uncertain" a model is in predicting a new word given a context.

Source	Weight	Word Count	Domain
BNC	8%	800K	Dialogue
CHILDES (MacWhinney, 2000)	29%	2.9M	Dialogue, Child Directed
Project Gutenberg (Gerlach and Font-Clos, 2018)	26%	2.6M	Fiction, Nonfiction
OpenSubtitles (Lison and Tiedemann, 2016)	20%	2M	Dialogue, Scripted
Simple English Wikipedia	15%	1.5M	Nonfiction
Switchboard (Stolcke et al., 2000)	1%	100K	Dialogue

Table 1: Data sources and their respective weights, word counts, and domains.

Finally, [Martinez et al. \(2023\)](#) tried an approach called "*objective curriculum*" where they used a simplified variant of Masked Language Modelling (MLM). Standard MLM involves training the encoder-part of a transformer by hiding or "masking out" a token and have the language model predict the hidden token ([Devlin et al., 2019](#)). [Martinez et al. \(2023\)](#) simplified this approach by having the model only predict the part-of-speech tag of the hidden word, beginning with only Verbs and Nouns and then gradually increasing the number of word classes ([Martinez et al., 2023](#), pp.117).

A recent paper outside of BabyLM is [Eldan and Li \(2023\)](#) who pre-trained small decoder models of about 125 million parameters on a synthetic dataset of children's stories. They then had these models predict the continuation of a story. They introduced a new type of evaluation where they instructed GPT-4 to pretend to be a teacher and grade the output generated by the smaller models on a scale from one to ten within the categories of grammar, creativity, consistency and plot. They compared this with GPT-XL and in most cases the smaller models trained on children's stories generated better continuations.

3 Data and preprocessing

In this paper, I used a dataset of the same size as the BabyLM *strict-small* track (see Table 1). A significant part - 29 % consists of data from the CHILDES corpus ([MacWhinney, 2000](#)). The CHILDES corpus only made up 5% of the BabyLM challenge ([Warstadt et al., 2023](#), pp.5). I used this dataset for the baseline models.

For the models where I experimented with ESL-data, I used the Teacher Student Chatroom Corpus (TSCC) ([Caines et al., 2020, 2022](#)) instead of the CHILDES corpus.

The TSCC consists of written one-to-one lessons between an English teacher and an adult English

student. The corpus was collected at the University of Cambridge in 2020 and 2022. Two English teachers were hired to hold the online lessons and in total there were 13 different students from across 7 different native languages. The English level of the students ranged from B1 to C2 according to the Common European Framework of Reference for Languages (CEFR)¹. B1 corresponds to lower intermediate and C2 is advanced, so this means no complete beginners were represented in the corpus. The corpus was annotated with CEFR-level and the conversational turns were annotated according to Teacher feedback types, like "scaffolding" and "eliciting". The dataset included both the original text from the chatrooms, as well as a corrected version that included punctuation and correcting of morphological errors (i.e. miswritten words). The syntax was not corrected. According to the articles, the TSCC consists of 260 lessons, 41.400 conversational turns and 362.900 tokens in total. ([Caines et al., 2020, 2022](#)).

I decided to use the corrected version of the raw text for my experiments. I imported the dataset and extracted all the corrected text. I first manually inspected the dataset and did some preprocessing. The most important thing I did, was replace strings that consisted of one or more underscores with the <UNK> token. The reason these were in the corpus in the first place, was that teachers used them as cloze-tasks (fill in the blank) for the students. An example: "*Some parents make their kids do loads of lessons like violin, karate, extra maths etc. etc. They are really _____.*" The word the teacher is looking for the student to provide here is probably "*strict*" or less likely "*ambitious*".

¹<https://www.coe.int/en/web/common-european-framework-reference-languages/table-1-cefr-3.3-common-reference-levels-global-scale>

3.1 Backtranslation

After preprocessing, I was left with a file that consisted of 294.409 tokens. This was a problem considering that the CHILDES dataset consisted of 2.839.591 tokens - almost 10 times as many. I couldn't expect the models trained on the TSCC dataset to be able to compete when it had only a tenth of the input.

I decided to try to augment the TSCC data using backtranslation. I used the Helsinki-NLP/Opus-MT machine translation models (Tiedemann and de Gibert, 2023). I wanted to use frequent languages on the internet for backtranslation, as I expected their machine translation models to be better. I kept adding languages until I had enough tokens, and ended up using the following languages for backtranslation of the TSCC dataset: French, Spanish, Italian, Russian, Mandarin Chinese, Dutch, German, Finnish, Swedish and Danish. Some of them are both big and linguistically close to English, namely French, Spanish, Italian and German. Dutch, Danish and Swedish are smaller but similar to English. Russian and Mandarin Chinese are big, but not very similar to English. The outlier is Finnish which is neither a big language nor linguistically close to English. The main reason I included Finnish was that the Helsinki-NLP/Opus-MT is a model developed in Finland and I therefore expected it to do well. The criteria for selecting these languages for backtranslating was both to get good backtranslations, as well as some variation.

After I had finished backtranslations, I discovered some special symbols that hadn't been stripped. Some of these were: ♪, •, # and ¶. It might not have mattered for the pre-training of the transformer models if I would have kept these in. In fact, some of them like the note-symbol ♪ and the paragraph symbol ¶ might have provided extra context that could have made them slightly better. However, since I was to compare with models trained on the CHILDES corpus, I decided this might be unfair one way or the other, so I stripped away the special symbols.

In the end I took the set of the original translations and all of the backtranslations to remove possible duplicates. The original TSCC text data plus all the backtranslations left me with a file that consisted of 2.938.160 tokens, which is very comparable to the 2.839.591 tokens of the CHILDES dataset.

3.2 Backtranslation quality

I will briefly discuss the quality of the backtranslations of four conversational turns. I picked the turns from index 9 to 12 in the TSCC training set for Spanish, French, German, Mandarin Chinese and Finnish (see Table 2). The translation of these four conversational turns for all of the languages can be found in the Appendix A - Table 18. From these examples it looks like all of the backtranslations did reasonably well when it comes to morphology and syntax. All of the words are spelled correctly and the word order looks reasonable. There were however some problems regarding semantics. The text starts with the teacher praising the student for the results on an English reading, after which they then go on to speak about the results on a listening test. In the original text the student says: "*around 30, listening is harder for me.*" This is a case of ellipsis, where the number 30 refers to 30 "*points*". We see that French, Mandarin Chinese and Finnish interpret 30 as *age*, and German interpret it as *time*. All of these were wrong. In fact, the only language that managed to backtranslate this turn correctly was Spanish.

Backtranslation via Spanish provided the examples that were most similar to the source text, often resulting in near-identical sentences. This would be good from a translator's point of view, but for my use this was not ideal, since it resulted in less variability.

The backtranslations via Mandarin Chinese were quite different from the source text, but in many cases they were semantically wrong or had cut a lot of the content from the source text. There is one particularly bad translation: The text where the teacher comforts the student by saying: "*ok that's quite common to be better at reading - but 30 is still more than respectable. i guess maybe speaking is the toughest part?*" is backtranslated to *30 years of age isn't worthy of respect.*

Finnish was a language I was a little worried about, given that it's not very big and linguistically far from English, but it turned out to be a good choice. It looks like Finnish provided a little more variability than the Romance and Germanic languages, and did not do worse when it comes to preserving semantics. If I were to repeat this experiment, I would probably not have included Mandarin Chinese and instead explored if Estonian had the same qualities as Finnish.

English	Spanish	French	German	Mandarin Chinese	Finnish
Wow that's really good - well done... you've definitely made a lot of progress. and what's your normal score on listening these days?	Wow that's really good - well done... you've definitely made a lot of progress. and what's your normal score on listening these days?	You have certainly made a lot of progress, and what is your normal listening score these days?	This is really good - well done... You have definitely made a lot of progress. and what is your normal score when listening these days?	Wow, really good— well done... you've really made a lot of progress.	Wow, it's really good - well done... you've definitely made a lot of progress. And what's the normal result of listening these days?
Around 30, listening is harder for me.	Around 30, listening is harder for me.	About 30 years, listening is more difficult for me.	At 30 o'clock it's harder for me to listen.	I'm about 30 years old. I'm worse.	At about 30 years of age, listening is harder for me.
Ok that's quite common to be better at reading - but 30 is still more than respectable. I guess maybe speaking is the toughest part?	Ok that's quite common to be better at reading - but 30 is still more than respectable. I guess maybe speaking is the toughest part?	Ok it's pretty common to be better at reading - but 30 is even more than respectable. I guess speaking is perhaps the most difficult part?	Okay, that's pretty common to be better at reading - but 30 is still more than respectable. I guess maybe speaking is the most difficult part?	But 30 years of age isn't worthy of respect.	Okay, it's pretty common to be better at reading - but 30 is still more than respectable. Maybe talking is the hardest part?
Yeah, because it's difficult to practice...	Yes, because it's difficult to practice...	Because it's hard to train...	Yeah, because it's hard to practice...	Yes, because it's hard to practice...	Yeah, because it's hard to practice...

Table 2: Comparison of backtranslations for English, Spanish, French, German, Mandarin Chinese and Finnish

4 Training

The main point of my experiment was not to train the best possible tiny transformer-based model using ESL data, but rather to see if a model trained on ESL data could outperform a model trained on child-directed data. I trained each model three times, with the seeds: 42, 2024 and 5550. I did not perform a hyper-parameter search. For preprocessing, I concatenated the datasets, created tokenizers and segmented the training data. I then trained an encoder-model, a decoder-model and an encoder-decoder model with the parameters shown in table 3 for all the models.

The only difference is that the encoder-decoder model has listed 6 hidden layers and the other two has 12, but in practice this amounts to the same number of layers since the encoder-decoder is a dual model.

After training the baseline models, I substituted the processed CHILDES data for the combined original and backtranslated TSCC dataset. I followed the same preprocessing steps as for the original data, creating a separate tokenizer for the new training data. I then trained the three different models on the new training data with the new tokenizer. The models can be found on my github².

²http://github.uio.no/erlingmi/IN5550_exam/

5 Evaluation

I now had six models: three baseline models where 29% of the training data came from the CHILDES-corpus and three models where the CHILDES-corpus had been substituted with TSCC-data augmented by backtranslation. Each of these models had been trained three times on the same three seeds, which amounted to a total of 18 checkpoints. I evaluated each checkpoint using the BLiMP and LAMBADA metrics. I then averaged the results for each of the models, and these can be found in Table 4 and Table 5. More detailed results can be found in the appendix A.

I will describe each of the metrics below and comment on the performance of my six models.

5.1 The Benchmark of Linguistic Minimal Pairs: BLiMP

The The Benchmark of Linguistic Minimal Pairs (BLiMP) consists of a set of artificially generated minimal pairs. Each pair consists of an acceptable and an unacceptable example and the goal of the LM to be tested is to select, that is assign the higher probability, to the acceptable example. The BLiMP metric tests for twelve different linguistic phenomena (see Table 4). An example of an acceptable example of Determiner-Noun agreement could be "*She had bought that chair*", whereas an unac-

Table 3: Hyperparameters

Parameters	
attention_dropout_prob	0.1
attention_probs_dropout_prob	0.1
classifier_activation_function	gelu
classifier_dropout_prob	0.1
embedding_dropout_prob	0.1
ffn_activation_function	geglu
ffn_dropout_prob	0.1
hidden_size	192
intermediate_size	512
norm_eps	1e-05
max_position_embeddings	128
norm_type	layer_norm
num_attention_heads	3
num_hidden_layers	12 / 6
position_bucket_size	32
positional_embeddings	relative
post_norm	true
pre_norm	true
qk_size	192
v_size	192
vocab_size	3072
relative_embedding_dropout	0.1

ceptable example would be "*She had bought that chairs*" (Warstadt et al., 2020, pp.380). Accuracy is calculated for each of the twelve phenomena, as well as the average accuracy. That means if a model gets an accuracy of 50 it would be the same as randomly picking an example. When Warstadt et al. (2020) developed this benchmark in 2020 their best performing model was GPT-2 with an average accuracy of 81.5. GPT-2 has 36 layers, 774 Million parameters and was trained on approximately 8 Billion tokens (Warstadt et al., 2020, pp.382). To recap, all of my models have less than 7 Million parameters, 12 layers and was trained on approximately 10 Million tokens.

When looking at the results from Table 4 the first observation we can make is that the decoder models do significantly better than the encoder and encoder-decoder models. This is to be expected. For left-to-right languages the decoder generates token by token from left to right. This is useful for capturing syntactic elements in analytic languages like English which relies heavily on a fixed word order. Encoder Models like BERT are useful for creating rich embeddings, but not so much for dis-

tinguishing small syntactic differences.

The second observation we can make is that the best model overall turned out to be the baseline decoder model trained on the CHILDES data with an average accuracy of 70.20, but only by a very small margin. The decoder model trained on the TSCC dataset got an average accuracy of 70.10.

NPI licensing is the linguistic phenomena where the CHILDES-decoder outperform the TSCC-decoder the most with accuracies of 61.01 and 57.55 respectively. NPI is short for Negative Polarity Items and refers to words like *any* and *ever* that are only permitted in certain clauses like negations and interrogatives (Warstadt et al., 2020, pp.381). Many languages, one of them Norwegian, do not have the same distinctions as English between Negative Polarity Items, like "*any*" and Positive Polarity Items, like "*some*". The Norwegian word for both of these would be "*noen*". For this particular linguistic phenomena it seems like it's a disadvantage to be an adult "pre-trained" on another a language that presumably have different NPI / PPI - distinctions.

The CHILDES-decoder also does significantly better on *anaphor agreement* with accuracies of 82.63 and 80.12 respectively. Anaphor agreement refers to agreement between a reflexive pronoun and it's antecedent (Warstadt et al., 2020, pp.381). An example of an acceptable sentence would be "*He washed himself*" and an unacceptable sentence would be "*He washed him*". These kind of constructions can be confusing for a foreign student of English. In English these reflexive pronouns are gendered, but in many languages they are not. In Italian for instance the same sentence would be: "*(Lui) si è lavato*" for the male form and "*(Lei) si è lavata*" for the female form. Since Italian is a pro-drop language, hence the parenthesis around the subject pronouns, in most cases the only way to tell the difference in gender between these two sentences would be the ending of the past participle: "lavato/a". In fact, it would also be correct to say "*Lui/lei ha lavato/a se stesso/a*", where the reflexive pronoun changes from "*si*" to "*se*". The point here is not to go into an extended discussion of Italian syntax, but to emphasize that the reflexive pronoun in other languages may change in other situations and for different reasons than what is the case for English.

The TSCC decoder does better on *irregular forms* than the CHILDES decoder, with accuracies of 92.35 and 91.55 respectively. This is expected

Table 4: Blimp Average Results

Model	Encoder - CHILDES	Encoder - TSCC	Decoder - CHILDES	Decoder - TSCC	Encoder-decoder - CHILDES	Encoder-decoder - TSCC
Filler gap dependency	63.41	71.43	68.90	67.34	60.09	60.74
Subject verb agreement	49.52	51.40	74.30	76.12	55.47	53.69
NPI licensing	36.14	48.64	61.01	57.55	35.44	32.27
Control and raising	53.47	46.25	64.15	63.86	59.64	59.21
Argument structure	52.73	52.02	71.48	71.52	56.96	55.95
Ellipsis	30.17	38.30	72.00	70.92	61.32	49.63
Determiner-Noun agreement	50.50	50.70	81.39	80.68	69.77	62.85
Anaphor agreement	61.62	59.72	82.63	80.12	65.35	58.18
Binding	52.22	55.51	72.79	74.21	66.10	64.92
Island effects	34.60	26.00	51.61	54.40	45.34	44.46
Quantifiers	43.26	72.51	79.50	78.93	65.84	67.88
Irregular forms	49.18	53.80	91.55	92.35	81.55	72.10
Average accuracy	48.32	51.40	70.20	70.10	57.84	55.48

Table 5: Lambada – average results

Metric	Encoder - CHILDES	Encoder - TSCC	Decoder - CHILDES	Decoder - TSCC	Encoder-decoder - CHILDES	Encoder-decoder - TSCC
Accuracy	0.17	0.83	8.63	7.06	1.08	0.69
Perplexity	1453.86	531.95	21.48	21.04	52.42	137.62

since children usually learn the regular patterns first, especially when it comes to verbs, and only later in their linguistic development learn the irregular exceptions (Pinker, 1994, pp.278-279). Even though this is the case, some of the adult language learners represented in the TSCC corpus might also struggle with this. The languages most different from English, were Japanese, Mandarin Chinese and Thai (Caines et al., 2020, pp.12). Of these, I only have personal experience with Chinese which I know have a much simpler inflection than English. For these students, the irregular forms, in particular, might cause problems. The reason why the TSCC decoder still outperformed the CHILDES decoder could be that some of the students spoke pro-drop languages like Spanish and Italian that have a more complex inflection than English, and would be expected to do better than native-speaking children on verb inflection. This was indeed the case, which we can also observe from the linguistic phenomena of *subject-verb agreement* where the CHILDES-decoder had an accuracy of 74.30 and the TSCC-decoder had an accuracy of 76.12. We might hypothesize that if we filtered out the Asian

students and only kept the Romance-speaking ones, this difference would be even higher for both *irregular forms* and *subject-verb agreement*.

I should also mention that none of the models do bad when it comes to irregular forms. The much bigger GPT-2 decoder-model had an accuracy of 84.1 on irregular forms, which is worse than both of my decoder models (Warstadt et al., 2020, pp.382).

Another linguistic phenomena where the TSCC-decoder outperforms the CHILDES-decoder is *binding*, with accuracies of 74.21 and 72.79 respectively. Binding refers to the structural relationship between a pronoun and its antecedent, typically a noun (Warstadt et al., 2020, pp.381). Warstadt et al. (2020, pp.380) gives the following sentence as an acceptable example of binding: "*Carlos said that Lori helped him*". and this sentence as an unacceptable example: "*Carlos said that Lori helped himself*". It seems like binding is a linguistic phenomena that requires a sense of logic that might not be as developed amongst children. It could also be one of the phenomena that do not vary so much across languages, which means the adult language learners receive the benefit and not the penalty of

already knowing another language.

The third observation we can make, is that even though my TSCC models lost to the CHILDES models both for the decoder-model and the encoder-decoder model, it still won overall if we calculate the average accuracies for the three models: 58.79 for CHILDES and 58.93 for TSCC. The reason for this is that the TSCC encoder model did over 3 percentage points better than the CHILDES encoder model. It is not clear to me why this is the case, but one hypothesis could be that intermediate and advanced adult language learners, and especially their English teachers, would use a bigger and more varied vocabulary than children and their parents in an everyday setting. In fact, challenging students to expand their vocabulary would be one of the goals of the teachers. Parents do so too when speaking with children, but probably not as often and as systematically. So my hypothesis is that the encoder model in particular might have benefited from this higher type to token ratio. It is unclear how backtranslation might have affected the result. On the one hand we might have gotten some synonyms "for free", but on the other hand many of the sentences would be very similar.

5.2 Language Modelling Broadened to Account for Discourse Aspects: LAMBADA

The LAMBADA dataset is a next-word prediction task that requires a very broad context to get right. It was intentionally designed to be difficult for language models. [Paperno et al. \(2016, pp.1528\)](#) used crowdsourcing when selecting which examples to include in the dataset. They first tested examples on two separate human reviewers giving the entire context, typically about 5 sentences before the target sentence and only included the examples that both of them got right. Of the examples that passed this hurdle, *only* the target sentence - excluding the context - were given to ten new human reviewers and only if all of them got the next word prediction *wrong*, was the word included in the LAMBADA dataset. The aim of this selection process was to only pick examples where it was imperative to understand the broader context in order to be able to predict the next word correctly.

Most LM's don't do well on the LAMBADA metric. It is still an open question whether this metric actually tests for language understanding or if it simply tests for the ability to infer from a very long context. The LAMBADA results of

my models (see [Table 5](#)) mirror the BLiMP results, with the CHILDES decoder being the best with an accuracy of 8.63. The TSCC decoder was the second best model for this metric too, but it lags a little more behind than was the case for BLiMP, with an accuracy of 6.77. For the perplexity metric lower is better and here the TSCC decoder got the lower score of 21.04 compared to 21.48 for the CHILDES decoder.

The CHILDES Encoder was the worst with an accuracy of 0.17 and a perplexity of 1453.86. In fact for two of the seeds, it wasn't able to predict a single next-word correctly ([Table 7](#)).

6 Conclusion

The conclusion so far, must be that tiny LM's did not turn out to be better students than babies, but neither were they much worse. The best model was the CHILDES decoder, but it only beat the TSCC decoder by a very small margin. The results are not very confident. One big source of uncertainty is my use of backtranslation as data augmentation of the TSCC corpus. It would have been better to have a bigger ESL corpus. Another alternative would have been to train on only the original TSCC dataset and one tenth of the CHILDES dataset. It might have been useful for comparison, but this would mean using only about 7.3 million tokens (compared to 10 million) and I was worried the metrics would be too uncertain using so few tokens as input. Another source of uncertainty is the fact that the TSCC data were collected from online chat rooms designed for this purpose, whereas the CHILDES data were collected by audio clips from real life interactions and then transcribed. We use spoken and written language differently, so for comparisons sake it might have been better collecting spoken ESL data and transcribing them, but it wouldn't necessarily have resulted in better LM's, quite possibly the opposite.

7 Future work

Even though my results in this small experiment were not entirely convincing, I think my idea is good enough that it is worth testing further before being discarded. I looked for more ESL corpora, but apart from TSCC I couldn't find any that were easily accessible, open and large enough. In order to test my hypothesis further it would be necessary to get access to more ESL student-directed data. This doesn't have to be teacher-student conversa-

tions, but could be text from textbooks or transcriptions from educational films.

It would also be worth doing a more thorough hyperparameter search to find the best model for the different datasets. One possibility would be to replicate the Contextualiser approach that won the Loose track of the BabyLM challenge (Xiao et al., 2023, pp. 12-13).

Of the metrics used here, BLiMP mainly tests for the formal-linguistic aspects and LAMABADA for ability to consider long context. It might be beneficial to include at least some of the metrics from Super GLUE to test for the functional-linguistic aspects.

My hypothesis also needs a better theoretical foundation, both from a cognitive-linguistic perspective and from an NLP / Informatics perspective.

References

- Andrew Caines, Helen Yannakoudakis, Helen Allen, Pascual Pérez-Paredes, Bill Byrne, and Paula Buttery. 2022. [The teacher-student chatroom corpus version 2: more lessons, new annotation, automatic detection of sequence shifts](#). In *Proceedings of the 11th Workshop on NLP for Computer Assisted Language Learning*, pages 23–35, Louvain-la-Neuve, Belgium. LiU Electronic Press.
- Andrew Caines, Helen Yannakoudakis, Helena Edmondson, Helen Allen, Pascual Pérez-Paredes, Bill Byrne, and Paula Buttery. 2020. The teacher-student chatroom corpus. *arXiv (Cornell University)*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#). *Preprint*, arXiv:1810.04805.
- Emmanuel Dupoux. 2018. Cognitive science in the era of artificial intelligence: A roadmap for reverse-engineering the infant language-learner. *Cognition*, 173:43–59.
- Ronen Eldan and Yuanzhi Li. 2023. [Tinystories: How small can language models be and still speak coherent english?](#) *arXiv (Cornell University)*.
- Lucas Georges Gabriel Charpentier and David Samuel. 2023. [Not all layers are equally as important: Every layer counts BERT](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 238–252, Singapore. Association for Computational Linguistics.
- Martin Gerlach and Francesc Font-Clos. 2018. A standardized project gutenber corpus for statistical analysis of natural language and quantitative linguistics.
- Philip A. Huebner, Elior Sulem, Fisher Cynthia, and Dan Roth. 2021. [BabyBERTa: Learning more grammar with small-scale child-directed language](#). In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 624–646, Online. Association for Computational Linguistics.
- Philip A. Huebner and Jon A. Willits. 2021. [Chapter eight - using lexical context to discover the noun category: Younger children have it easier](#). In Kara D. Federmeier and Lili Sahakyan, editors, *The Context of Cognition: Emerging Perspectives*, volume 75 of *Psychology of Learning and Motivation*, pages 279–331. Academic Press.
- Daniel Jurafsky and James H. Martin. *Speech and language processing an introduction to natural language processing, computational linguistics, and speech recognition third edition draft*.
- 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).
- 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](#). *Preprint*, arXiv:1907.11692.
- Brian MacWhinney. 2000. *The CHILDES project : tools for analyzing talk : Vol. 1 : Transcription format and programs*, 3rd ed. edition, volume Vol. 1. Lawrence Erlbaum, Mahwah, N.J.
- Richard Diehl Martinez, Hope McGovern, Zebulon Goriely, Christopher Davis, Andrew Caines, Paula Buttery, and Lisa Beinborn. 2023. [CLIMB – curriculum learning for infant-inspired model building](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 112–127, Singapore. Association for Computational Linguistics.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv (Cornell University)*.
- Steven Pinker. 1994. *The language instinct*. Morrow, New York.
- David Samuel, Andrey Kutuzov, Lilja Øvrelid, and Erik Velldal. 2023. [Trained on 100 million words and still in shape: BERT meets British National Corpus](#). In

- Findings of the Association for Computational Linguistics: EACL 2023*, pages 1954–1974, Dubrovnik, Croatia. Association for Computational Linguistics.
- Julius Steuer, Marius Mosbach, and Dietrich Klakow. 2023. [Large GPT-like models are bad babies: A closer look at the relationship between linguistic competence and psycholinguistic measures](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 142–157, Singapore. Association for Computational Linguistics.
- Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. [Dialogue act modeling for automatic tagging and recognition of conversational speech](#). *Computational Linguistics*, 26(3):339–373.
- Jörg Tiedemann and Ona de Gibert. 2023. [The OPUS-MT dashboard – a toolkit for a systematic evaluation of open machine translation models](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 315–327, Toronto, Canada. Association for Computational Linguistics.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *Preprint*, arXiv:2307.09288.
- Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. [Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.
- Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohanney, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. [Blimp: The benchmark of linguistic minimal pairs for english](#). *Transactions of the Association for Computational Linguistics*, 8:377–392.
- Chenghao Xiao, G Thomas Hudson, and Noura Al Moubayed. 2023. [Towards more human-like language models based on contextualizer pretraining strategy](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 317–326, Singapore. Association for Computational Linguistics.
- Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. [Problems in current text simplification research: New data can help](#). *Transactions of the Association for Computational Linguistics*, 3:283–297.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. [Opt: Open pre-trained transformer language models](#). *Preprint*, arXiv:2205.01068.

A Appendix

Table 6: Blimp - Encoder CHILDES

Seed	42	2024	5550	Average
Filler gap dependency	71.43	71.43	47.37	63.41
Subject verb agreement	48.60	51.70	48.25	49.52
NPI licensing	29.60	47.63	31.20	36.14
Control and raising	57.48	46.98	55.96	53.47
Argument structure	54.01	50.04	54.14	52.73
Ellipsis	23.25	44.65	22.60	30.17
Determiner-Noun agreement	50.71	49.94	50.84	50.50
Anaphor agreement	65.60	46.90	72.35	61.62
Binding	52.39	57.17	47.10	52.22
Island effects	37.05	30.29	36.45	34.60
Quantifiers	32.65	71.50	25.62	43.26
Irregular forms	48.05	48.05	51.45	49.18
Average accuracy	48.44	51.29	45.22	48.32

Table 7: Lambada - Encoder - CHILDES

Seed	42	2024	5550	Average
Accuracy	0.00	0.52	0.00	0.17
Perplexity	1374.78	872.31	2114.50	1453.86

Table 8: Blimp - Encoder - TSCC

Seed	42	2024	5550	Average
Filler gap dependency	71.43	71.43	71.43	71.43
Subject verb agreement	51.75	51.82	50.62	51.40
NPI licensing	51.49	42.46	51.96	48.64
Control and raising	45.74	45.12	47.88	46.25
Argument structure	53.66	48.80	53.61	52.02
Ellipsis	40.50	37.35	37.05	38.30
Determiner-Noun agreement	51.02	50.29	50.80	50.70
Anaphor agreement	62.10	57.10	59.95	59.72
Binding	54.73	56.04	55.77	55.51
Island effects	26.36	27.19	24.44	26.00
Quantifiers	61.52	80.12	75.90	72.51
Irregular forms	51.30	54.95	55.15	53.80
Average accuracy	51.32	50.81	52.08	51.40

Table 9: Lambada - Encoder - TSCC

Seed	42	2024	5550	Average
Accuracy	2.19	0.07762	0.21	0.83
Perplexity	349.29	666.637	579.93	531.95

Table 10: Blimp - Decoder - CHILDES

Model	Decoder - CHILDES			
Seed	42	2024	5550	Average
Filler gap dependency	67.99	68.71	70.01	68.90
Subject verb agreement	75.30	73.62	73.98	74.30
NPI licensing	59.57	64.39	59.07	61.01
Control and raising	63.60	63.20	65.64	64.15
Argument structure	71.72	71.42	71.30	71.48
Ellipsis	71.90	71.45	72.65	72.00
Determiner-Noun agreement	81.42	79.64	83.10	81.39
Anaphor agreement	82.10	84.55	81.25	82.63
Binding	71.66	73.24	73.46	72.79
Island effects	49.12	54.43	51.28	51.61
Quantifiers	80.95	79.55	78.00	79.50
Irregular forms	92.70	88.15	93.80	91.55
Average accuracy	69.72	70.51	70.36	70.20

Table 11: Lambada - Decoder - CHILDES

Seed	42	2024	5550	Average
Accuracy	8.95	8.83	8.11	8.63
Perplexity	21.39	21.46	21.58	21.48

Table 12: Blimp - Decoder - TSCC

Seed	42	2024	5550	Average
Filler gap dependency	68.44	66.01	67.57	67.34
Subject verb agreement	76.35	76.85	75.17	76.12
NPI licensing	54.99	57.59	60.07	57.55
Control and raising	62.68	65.06	63.84	63.86
Argument structure	72.04	71.42	71.09	71.52
Ellipsis	70.85	69.15	72.75	70.92
Determiner-Noun agreement	80.58	81.69	79.77	80.68
Anaphor agreement	79.35	81.45	79.55	80.12
Binding	73.99	74.66	73.97	74.21
Island effects	50.81	54.71	57.67	54.40
Quantifiers	77.85	76.55	82.38	78.93
Irregular forms	88.10	95.20	93.75	92.35
Average accuracy	69.27	70.24	70.78	70.10

Table 13: Lambada - Decoder - TSCC

Seed	42	2024	5550	Average
Accuracy	6.77	6.62	7.80	7.06
Perplexity	21.48	21.43	20.22	21.04

Table 14: BLiMP - Encoder-decoder - CHILDES

Seed	24	2024	5550	Average
Filler gap dependency	59.44	59.99	60.84	60.09
Subject verb agreement	55.00	55.62	55.80	55.47
NPI licensing	40.11	30.67	35.53	35.44
Control and raising	61.14	58.26	59.52	59.64
Argument structure	55.87	58.17	56.83	56.96
Ellipsis	63.20	56.80	63.95	61.32
Determiner-Noun agreement	74.75	60.38	74.17	69.77
Anaphor agreement	69.80	56.45	69.80	65.35
Binding	66.39	66.64	65.27	66.10
Island effects	43.30	46.55	46.16	45.34
Quantifiers	62.95	75.33	59.23	65.84
Irregular forms	83.40	84.10	77.15	81.55
Average accuracy	58.64	56.73	58.16	57.84

Table 15: Lambada - Encoder-decoder - CHILDES

Seed	42	2024	5550	Average
Accuracy	0.17	1.94	1.14	1.08
Perplexity	68.55	42.70	46.00	52.42

Table 16: BLiMP - Encoder-decoder - TSCC

Seed	24	2024	5550	Average
Filler gap dependency	60.24	58.84	63.14	60.74
Subject verb agreement	53.83	54.75	52.50	53.69
NPI licensing	29.07	35.79	31.94	32.27
Control and raising	60.12	59.94	57.58	59.21
Argument structure	56.71	55.24	55.91	55.95
Ellipsis	63.40	46.65	38.85	49.63
Determiner-Noun agreement	64.49	65.05	59.00	62.85
Anaphor agreement	63.20	56.45	54.90	58.18
Binding	64.41	65.43	64.93	64.92
Island effects	44.57	40.77	48.04	44.46
Quantifiers	66.42	69.08	68.15	67.88
Irregular forms	76.80	68.85	70.65	72.10
Average accuracy	56.05	55.41	54.98	55.48

Table 17: Lambada - Encoder-decoder - TSCC

Seed	42	2024	5550	Average
Accuracy	2.04	0.02	0.0	0.69
Perplexity	43.47	139.02	230.36	137.62

English	French	Spanish	Italian	Russian	Mandarin Chinese
Wow that's really good - well done... you've definitely made a lot of progress. and what's your normal score on listening these days?	You have certainly made a lot of progress, and what is your normal listening score these days?	Wow that's really good - well done... you've definitely made a lot of progress. and what's your normal score on listening these days?	Wow that's really good - well done... you definitely made a lot of progress. and what's your normal score listening these days?	Wow, that's really good- well done- you've certainly made great progress, and what's your usual bill for listening these days?	Wow, really good- well done... you've really made a lot of progress.
Around 30, listening is harder for me.	About 30 years, listening is more difficult for me.	Around 30, listening is harder for me.	About 30 years, listening is harder for me.	About 30, it's harder for me to listen.	I'm about 30 years old. I'm worse.
Ok that's quite common to be better at reading - but 30 is still more than respectable. I guess maybe speaking is the toughest part?	Ok it's pretty common to be better at reading - but 30 is even more than respectable. I guess speaking is perhaps the most difficult part?	Ok that's quite common to be better at reading - but 30 is even more than respectable. I guess maybe talking is the hardest part?	Ok that's common enough to be better at reading - but 30 is even more than respectable. I guess maybe talking is the hardest part?	It's good that it's quite often better to read - but 30 is still more than respectable.	But 30 years of age isn't worthy of respect.
Yeah, because it's difficult to practice...	Because it's hard to train...	Yes, because it's difficult to practice...	Yeah, 'cause it's hard to practice...	Yes, because it's hard to train...	Yes, because it's hard to practice...
English	Dutch	German	Finnish	Swedish	Danish
Wow that's really good - well done... you've definitely made a lot of progress. and what's your normal score on listening these days?	Oh, wow that's really good - well done... you've certainly made a lot of progress. And what's your normal score on listening these days?	This is really good - well done... You have definitely made a lot of progress. and what is your normal score when listening these days?	Wow, it's really good - well done... you've definitely made a lot of progress. And what's the normal result of listening these days?	Wow it's really good - well done... you've definitely made a lot of progress. and what's your normal point on listening these days?	Wow it's really good - well done... you've definitely made a lot of progress. And what's your normal score on listening these days?
Around 30, listening is harder for me.	About 30 o'clock listening is harder for me.	At 30 o'clock it's harder for me to listen.	At about 30 years of age, listening is harder for me.	About 30, listen is harder for me.	About 30 years, it's harder for me to listen.
Ok that's quite common to be better at reading - but 30 is still more than respectable. I guess maybe speaking is the toughest part?	OK that's very common to be better at reading - but 30 is still more than respectable. I think speaking is the hardest part?	Okay, that's pretty common to be better at reading - but 30 is still more than respectable. I guess maybe speaking is the most difficult part?	Okay, it's pretty common to be better at reading - but 30 is still more than respectable. Maybe talking is the hardest part?	OK it's pretty common to be better at reading - but 30 is still more than respectable. I guess speaking is the toughest part?	Well, it's quite common to be better at reading - but 30 is still more than respectable. I guess speeches are the hardest part?
Yeah, because it's difficult to practice...	Yes, because it's hard to practice...	Yeah, because it's hard to practice...	Yeah, because it's hard to practice...	Yes, because it's hard to train...	Yeah, because it's hard to practice...

Table 18: Comparison of backtranslations for all of the languages

Parameter Sharing for Tiny Language Models

Eirik Eggset
University of Oslo

Rayyan A. S. Syed
University of Oslo

Torkild Finne
University of Oslo

Abstract

In recent years the trend has been that language models (LMs) has grown in size. Some of the challenges faced when using these are GPU capacity and long training times. This is motivation for aiming to reduce the model size. This paper is based on the TinyLM track(Charpentier, 2024), where the goal is to improve small language models with a max size of 7 million parameters. Our approach is based on the layer-wise parameter sharing as proposed in (Takase and Kiyono, 2023). They proposed three assignment rules: SEQUENCE, CYCLE, and CYCLE (REV), for deciding which layers are assigned the same parameters. Their results show that this approach gives improvements in terms of the parameter size and computational time for relatively small models. Motivated by these results, we aim to see if these improvements translate well to much smaller models as well. We observe some minor improvements in some setups. It also showed potential for improved results for future experiments, and perhaps for downscaling larger models.

1 Introduction

Language models (LMs) have progressively increased in size, presenting challenges such as limited GPU capacity and extended training times. Smaller models contain less trainable parameters, which leads to faster and cheaper training. The drawback is that larger models usually outperforms smaller ones. Therefore it can be beneficial to try to reduce the model size, without degrading the model performance.

This paper is based on the "Pretraining of Tiny Language Models" track(Charpentier, 2024). This track is inspired by the annual BabyLM challenge(Warstadt et al., 2023), where the goal is to train a language model on a small dataset. Similarly to the BabyLM challenge, the goal of this

track is to pretrain a language model and to investigate potential improvements compared to predetermined benchmarks. In contrast to this challenge, the track does not limit the training data, but the model's number of trainable parameters. The set limitations are a maximum of 7 million parameters, and a maximum of 5 hours of training time. The track uses two different metrics as a benchmark, namely LAMBADA(Paperno et al., 2016) and BLiMP(Warstadt et al., 2020).

The main motivation for our experiments is the research done by Takase and Kiyono in (Takase and Kiyono, 2023). They tested three different ways to reuse transformer layers in transformer models. The main idea is that by reusing the same parameters, you can increase the size or number of the layers and still keep a similarly sized model.

Takase and Kiyono focused on stacking multiple smaller layers, resulting in a deeper model. They compared their results to another approach for parameter sharing used in the Universal Transformer architecture(Dehghani et al., 2018). Here, instead of adding multiple small layers to compensate for the now fewer trainable parameters, the existing weight matrices are increased. Even though the paper showed promising results of increasing the expressive power over classical transformer models, increasing the weight matrices is slower than adding more small layers(Takase and Kiyono, 2023).

This paper will document our experiments and results for parameter sharing for tiny language models. In Section 2 we will go through the background theory and the motivation for our choices. In Section 3 we will go through our methodology. In Section 4 we will present our results. In the sections 5 and 6 we will discuss and conclude our findings.

2 Background

2.1 Transformer models

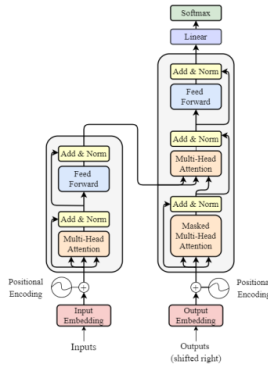


Figure 1: Encoder-decoder architecture(Vaswani et al., 2017)

The model we are trying to improve is based on the transformer architecture, as proposed in (Vaswani et al., 2017). It is divided in two individual parts, an encoder and a decoder. In the original paper, the encoder output is linked to the decoder through a concept called cross-attention. However, in later years it has been popular to either use an encoder-only or decoder-only architecture(Devlin et al., 2018; Samuel et al., 2023; Radford et al., 2018, 2019).

As illustrated in Figure 1, the encoder comprises N identical layers, each containing a self-attention mechanism and a feed-forward network. The difference between an encoder-only and decoder-only model is the masking of future tokens. While an encoder model is trained to predict a random token within a sequence(Devlin et al., 2018), the decoder is trained in an auto-regressive fashion, meaning it will predict the last token in a sequence(Radford et al., 2018).

Transformer models uses a concept called attention to capture context from surrounding tokens(Vaswani et al., 2017). Attention utilizes query, key and value vectors to model dependencies across the input sequence. This allows for efficient and parallelized computations.

A transformer model takes tokenized text as input. In the baseline model the text is converted using BytePiece(Wu et al., 2016) tokenization. This algorithm uses a hyperparameter called merge operations in order to determine the vocabulary size, which is not well understood(Gowda and May, 2020). However it is reported that different vocabulary sizes may affect the performance(Martinez et al., 2023).

2.2 Metrics

Measuring the effectiveness of a language model can be challenging, as there is often multiple correct answers. To evaluate the performance of our models, we used the metrics LAMBADA(Paperno et al., 2016) and BLiMP(Warstadt et al., 2020). These are commonly used methods for evaluating language models(Devlin et al., 2018; Warstadt et al., 2023; Charpentier, 2024)

2.2.1 BLiMP

To assess the models' grammatical knowledge, we utilized the BLiMP (Benchmark of Linguistic Minimal Pairs) dataset(Warstadt et al., 2020). This dataset comprises pairs of sentences that contrast incorrect and correct syntax, morphology, or semantics, providing a clear basis for evaluation. The metric tests the models' performance on different categorized phenomena.

BLiMP consists of 67 automatically generated datasets of 1000 minimal pairs each. These are divided into 12 phenomena. They are categorized in multiple groups, where each looks at it's own grammatical rules. We have added two examples in Table 1.

2.2.2 LAMBADA

LAMBADA (LAnguage Modeling Broadened to Account for Discourse Aspects) tests whether the model can understand the context within a broader discourse (Paperno et al., 2016). The idea behind LAMBADA is that one should be able to predict the last word of a sequence if given the entire passage, but not if only the last sentence is provided. This implies that a model excels in LAMBADA if it can maintain awareness of the entire passage, rather than mainly focusing on the local context. The model is tasked to guess the last word of a context sequence.

The dataset consists of 5,153 passages with a context sentence, a target sentence and a target word each. The context sentences is provided to define the target word from the target sentence. The target sentence is built up by a sentence where the last word is masked. A complete example passage from the dataset is shown in figure 2.

The metric itself gives an accuracy score as well as a perplexity score. The accuracy tells us how many of the words were predicted correctly. The perplexity tells us how sure the model is about its prediction. A model with low perplexity usually

Phenomenon	Acceptable Example	Unacceptable Example
ANAPHOR AGR	Many girls insulted themselves .	Many girls insulted herself .
ARG. STRUCTURE	Rose wasn't disturbing Mark.	Rose wasn't boasting Mark.

Table 1: Example of BLiMP dataset pairs with acceptable and unacceptable sentences.

indicates that the model is more confident, and the answer is less based on random choice.

Context: “Yes, I thought I was going to lose the baby.” “I was scared too,” he stated, sincerity flooding his eyes. “You were?” “Yes, of course. Why do you even ask?” “This baby wasn’t exactly planned for.”
Target sentence: “Do you honestly think that I would want you to have a ——?”
Target word: miscarriage

Figure 2: LAMBADA Context and Target Sentence with Target Word

2.3 Dataset

Studies such as CLIMB (Martinez et al., 2023) highlights infants’ sensitivity to linguistic distributional aspects and curriculum learning. Martinez proposes a learning framework similar to an infant’s progression from basic word memorization to recognizing lexical categories.

We used the 10 million words dataset provided by (Charpentier, 2024). This consists of 6 different corpora. This dataset uses a lot of the same corpora as in CLIMB, however it is only a small subset and the order of training data is randomized. The fractions used of each dataset can be found in table 2.

The SWITCHBOARD corpus (Godfrey et al., 1992), comprises of conversational speech and text from roughly 500 speakers across the U.S. Open-Subtitles collection (Lison and Tiedemann, 2016) contains movie and TV subtitles and provide parallel corpora in over 60 languages.

The CHILDES Project (MacWhinney, 2000) is a datasets with conversational interactions from monolingual and bilingual children, caregivers, and individuals with language disabilities.

The Standardized Project Gutenberg Corpus (SPGC) (Gerlach and Font-Clos, 2020) include over 50,000 books and 3 billion word-tokens.

Some part of the BNC (British National Corpus) (Consortium et al., 2007) was also included. This data set consist of a wide range of newspapers,

journals, books, letters, essays, unscripted informal conversations, radio shows and phone calls from the late 20th century.

3 Method

3.1 Parameter sharing

The different techniques we used for parameter sharing was the same as in (Takase and Kiyono, 2023). They proposed three different ways to configure which layers share parameters, which they called SEQUENCE, CYCLE and CYCLE (REV). These where the three techniques we chose to test out. We refer to these as assignment strategies, as illustrated in Figure 3.

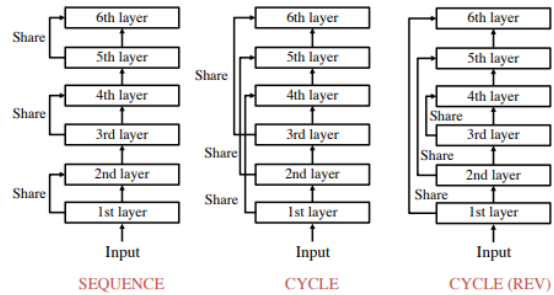


Figure 3: Parameter sharing strategies (Takase and Kiyono, 2023)

When implementing the parameter sharing, we created a model with a total depth N and M unique layers. With layer we refer to either an encoder-layer or a decoder-layer, consisting of both attention-heads as well as a feed forward network. In our implementation we add the number of shared parameter layers in the configuration file. This value is used alongside the total number of layers to calculate M .

In the forward function of the encoder and decoder models, we use a special "assignment strategy function", which takes an index and returns the corresponding layer based on the selected assignment strategy. When initializing the model we only create M unique layers, indexed by j . During training and inference, we iterate over all N layers with the layer index i . The chosen assignment strategy

Source	Weight	Word Count	Domain
BNC	8%	800K	Dialogue
CHILDES	29%	2.9M	Dialogue, Child Directed
Project Gutenberg	26%	2.6M	Fiction, Nonfiction
OpenSubtitles	20%	2M	Dialogue, Scripted
Simple English Wikipedia	15%	1.5M	Nonfiction
Switchboard	1%	100K	Dialogue

Table 2: Source data used in the project with their respective weight, word count, and domain.(Charpentier, 2024)

function is used to map i to its corresponding value j .

Each assignment strategy function has a guard clause which checks if the index exceeds the number of shared layers. If it does, it starts to index incrementally from after the last shared layer. This ensures that all layers after the first parameter sharing ones gets it’s own unique parameters. If the number of shared layers is set to 0, it effectively becomes an identity function, which always returns the input.

Our implementation also keeps the last layers without parameter sharing, unless specified, meaning they have their own unique parameters. The assignment strategies, which are displayed in Algorithms 1, 2 and 3. In the algorithms, the value "num_shared_layers" tells us how many of the N layers use parameter sharing. The parameter sharing layers are always at the start.

For example if you have an 8 layer deep model, where the first 6 layers use parameter sharing, we would iterate over the indexes from 0 to 7. By sending the numbers 0, 1, 2, 3, 4, 5, 6, 7 into the function sequence selection 1 with parameter sharing in the first 6 layers, the output would respectively be 0, 0, 1, 1, 2, 2, 3, 4. In this configuration there would only be initialized 5 unique layers, where 3 are used for the first 6 parameter sharing layers, and the last two are unique.

Takase and Kiyono used an encoder-decoder model. From their testing, parameter sharing seemed to improve the model in terms of the parameter size and computational time, for language translation tasks. Our goal is to investigate whether similar results can be seen for our task.

Algorithm 1 Sequence Selection

```

function SEQUENCESELECTION
  if  $i \geq numSharedLayers$  then
    return  $i - \lfloor \frac{numSharedLayers}{2} \rfloor$ 
  else
    return  $\lfloor \frac{i}{2} \rfloor$ 
  end if
end function

```

Algorithm 2 Cycle Selection

```

function CYCLESELECTION
   $midpoint \leftarrow \lfloor \frac{totalUniqueLayers}{2} \rfloor$ 
  if  $i \geq numSharedLayers$  then
    return  $i - \lfloor \frac{numSharedLayers}{2} \rfloor$ 
  else if  $i < midpoint$  then
    return  $i$ 
  else
    return  $i - midpoint$ 
  end if
end function

```

Algorithm 3 Cycle Reverse Selection

```

function CYCLEREVERSESELECTION
   $midpoint \leftarrow \lfloor \frac{numSharedLayers}{2} \rfloor$ 
  if  $i \geq numSharedLayers$  then
    return  $i - midpoint$ 
  else if  $i < midpoint$  then
    return  $i$ 
  else
    return  $numSharedLayers - 1 - i$ 
  end if
end function

```

3.2 Baseline models

The two baseline models we used was the default as was provided by (Charpentier, 2024). The two architectures we investigated was the Encoder-only and Decoder-only architectures.

The encoder models was trained using Masked Language Modeling such as in (Devlin et al., 2018). The decoder was trained using causal masking, such as in (Radford et al., 2018, 2019).

Both the encoder and the decoder has the same default configuration values. The values we tried scaling up can be found in Table 3, along with their baseline values.

The baseline encoder and decoder was approximately 6 million parameters. Given that the task was to train a model below 7 million we also compared the parameter sharing models to a baseline where the parameters were scaled up to right below 7 million parameters. This scaling was done in the same way as our deep-wide models.

Because there always is a degree of randomness when evaluating a trained model, it would not be fair to only train a single baseline model and compare it to a huge number of other configurations. In order to minimize this disadvantage we trained multiple baseline models, both scaled and not, and chose the best performing metrics for each.

Parameters	Values	Scaling
hidden_size	192	Width
intermediate_size	512	Width
qk_size	192	Width
v_size	192	Width
num_hidden_layers	12	Depth

Table 3: Model Configuration Parameters, their default value and scaling dimension

3.3 Configurations and Scaling

All the models used the exact same configuration as the baseline models, except for the number of shared layers and the values in Table 3 which was scaled up.

Throughout this section we refer to scaling in depth and width. With depth we mean increasing the number of layers in the model, similarly to what was done in (Takase and Kiyono, 2023), while width refers to increase the size of the hidden and output layers of the feed forward network at the end of the block, as in (Dehghani et al., 2018). For some of the models where we used more than 6 shared layers, we also increased the query, key and value vector sizes from 192 to 256, and therefore also the number of attention heads from 3 to 4. The column **Scaling** in Table 3 shows which values we considered when scaling in a certain dimension.

When introducing parameter sharing between

two layers, the number of parameters used by both layers is halved. To get back to the original parameter count, there are multiple ways to scale up the model (Takase and Kiyono, 2023; Dehghani et al., 2018). For our experiments we tested four different approaches for re-scaling. This included width, depth, vocabulary size and a combination of the prior approaches. We also tried different number of layers which shared parameters. This is because by having more sharing layers, we can scale even more in the depth or width.

First we wanted to do a simple investigation of how introducing parameter sharing in a model affected its performance, without changing the configuration values. This was so we could compare the effects parameter sharing has on down scaling versus up scaling of a model.

If we were to test every single combination it would take a lot of time and resources. Instead we ran multiple tests in multiple "training rounds", where we eliminated the worst configurations for each round.

For the first round we tested scaling in the width versus scaling in depth. The specific configurations included one scaled in width, one scaled for depth and one scaled a smaller amount in both width and depth. For each of these configurations we tested for all three assignment strategies. We also tested for three different levels of pairwise parameter sharing. One where we only shared in the first four layers, one where we shared in the 10 first layers and finally one where all layers shared. For all three the parameter sharing was done in pairs of two, so each selected layer shared with exactly one other layer.

In the next round we picked the best configurations and tested for 25,000 steps. We also experimented with scaling in both depth and width, but more weighted towards one of the two directions. This weighting was based on the results from the first round.

For the third round we tested the best performing models with a vocabulary size of 6144. We tested for multiple numbers of parameter sharing layers. Here we also tested for all three assignment strategies, in case there is a difference for the vocabulary size.

For the final round we tested a variety of different configurations we thought was interesting, but did not fit into the earlier round. This includes different numbers of shared layers, and different combinations of configuration values.

4 Results

For the experiments, we have generated baselines for both the decoder and encoder to compare the results. Additionally, we have generated and included metrics for the baselines, for both 10,000 and 25,000 steps to assess potential scaling improvements in the experiments. As we are working with small languages models, we were able to generate many different models with different setups. Since there were a lot of different and comprehensive results we will not include everything. This section includes the best and most interesting results from our experiments.

4.1 Cost of parameter sharing

Our main study focuses on experimenting with parameter sharing. As described in Section 3.3, the strategy is to first downscale our model with parameter sharing before upscaling it to a corresponding parameter count. In Table 4, we present the cost of only implementing parameter sharing for the decoder. We have also performed the downscaling for the encoder, but the results appeared to be mostly random. Therefore, we have chosen not to include them. In the following subsections, we will explore if scaling the configuration values up again yields any improvement.

As seen in Table 4, the performance of all models decreases compared to the baseline. However, this is expected due to the lower parameter count. The interesting aspect of the results is how much the model’s performance decreases and for which architecture.

Group-wise, our different parameter sharing architectures have the same size for the same number of shared parameters. As shown in Table 4, the cycle reverse strategy outperforms the sequential and cycle strategies. Our results support the findings of the (Takase and Kiyono, 2023), where cycle reverse is indicated to be the best of the three architectures.

4.2 BLiMP

4.2.1 Results from the Encoder

The results from the encoder were poor. An overview of our results can be found in Table 5. Out of the assignment strategies cycle reverse was the most successful. Therefore we have chosen to display these in the table.

For the models trained for 10,000 steps most of the results had an average accuracy of around 50. The best performing model was the model with 6

Decoder	Size	Acc.	Perp.	BLiMP
Baseline	5.97M	7.93	21.96	71.02
6 Seq	4.63M	7.70	23.94	67.52
6 Cycle	4.63M	6.64	24.45	68.37
6 Cycle rev	4.63M	6.58	24.07	68.77
12 Seq	3.30M	5.65	26.29	66.54
12 Cycle	3.30M	4.56	26.9	66.73
12 Cycle rev	3.30M	6.66	24.82	67.92
18 Seq	1.97M	3.70	30.71	63.68
18 Cycle	1.97M	3.67	31.30	63.09
18 Cycle rev	1.97M	4.31	29.94	64.72

Table 4: Decoder performance metrics for LAMBADA accuracy and perplexity, and BLiMP average accuracy for 10,000 training steps.

shared layers which had increased intermediate and hidden sizes. This model reached an average accuracy of 52.78. The second best was the baseline, right behind with a score of 52.20.

Most of the BLiMP scores did for some reason decrease after running for 25,000 steps. Surprisingly our scaled baseline was outperformed by the un-scaled baseline for both 10,000 and 25,000 steps.

The number of shared layers did not seem to affect the results too much, as the scores were spread quite evenly throughout for both 10,000 and 25,000 steps.

4.2.2 Results from the Decoder

All our decoder models outperformed the best encoder model.

The best parameter sharing decoder model did not outperform the decoder baseline model for 10,000 steps, which scored 71.02 as seen in Table 5. One of our models however, did score the same as the baseline.

It seemed like the parameter sharing models gained more ground on the baseline decoder score, where one even outperformed it with 71.77 for 25,000 steps. Here the baseline model stayed approximately the same, however most of the other models increased their accuracy slightly.

4.3 LAMBADA

4.3.1 Results from the Encoder

In general, our encoder models produced poor results across the board. None of the experimental models demonstrated any significant improvements. As seen in Table 6, most of the scores are either 0 or close to 0 for the 10,000 steps. The best-performing

Config	Shared layers	Dec 10k	Enc 10k	Dec 25k	Enc 25k
Baseline	0	71.02	52.20	71.09	47.58
Baseline scaled	0	70.10	51.53	70.71	48.41
Deep cycle rev	6	69.31	50.38	70.60	50.06
Wide cycle rev	6	66.89	52.78	70.06	50.24
Wide cycle rev	8	70.05	43.98	69.24	45.35
Deep cycle rev	10	69.84	51.04	70.15	51.43
Wide/deep cycle rev	12	68.40	51.71	70.72	49.97
Deep cycle rev	24	71.02	48.62	71.77	51.85

Table 5: Model performance for the BLiMP metric across different configurations and training steps.

model has a score of 0.19. This represents 10 correct predictions out of 5,153 different passages, in contrast to the baselines 0 (Paperno et al., 2016).

Although this indicates some improvement for our model, 0.19 is still very low. A random word from the context passage would result in a score of 1.6% (Paperno et al., 2016). Table 6 also shows that the models have very high perplexity, indicating that the results are quite random.

At 25,000 steps, most of the provided encoder models improve. The deep cycle reverse model outperforms the baseline model. However, the scaled baseline appeared on top with 2.74, but with a higher perplexity. This indicates that parameter sharing might offer some improvements for scaling.

4.3.2 Results from the Decoder

Unlike our encoder model, the decoder yielded some notable results on the LAMBADA metric. As the table shows, our best baseline decoder over multiple runs of 10,000 steps scored an accuracy of 8.3. None of the parameter sharing models came close. The second best was the scaled baseline with 8.0.

In general, we can see that most of the results are quite similar at 10,000 steps. However it seems like the parameter sharing might scale better than our baseline. The best model for 25,000 steps resulted in a score of 9.84 compared to our baseline model of 8.93. For 25,000 steps multiple parameter sharing models beat the baseline models.

4.4 Training time

The focus of this study was the performance on the metrics. However we also noticed a small change in the training time, as shown in Table 7. The wide models was slightly faster than the baselines, but the difference was marginal. The deep models were substantially slower.

Model	Time (min)
decoder baseline	29.58
decoder deep	39.92
decoder wide	29.47
encoder baseline	40.32
encoder deep	58.12
encoder wide	38.33

Table 7: Model training times for the baseline, deepest and widest models. Trained for 10,000 steps.

5 Discussion

5.1 Encoder vs Decoder

The decoder-based model’s superior performance compared to the encoder models might be attributed to its auto-regressive decoding strategy. This approach generates each token based on previously generated tokens, ensuring a coherent sequence output, which can in turn allow the model to learn a better syntactic representation than the encoder. The random masking in an encoder is not sequential by nature, which might be a reason for it to be outperformed.

The decoder-based models’ superior performance compared to the encoders’ might be attributed to the evaluation metrics. Decoders might naturally have an advantage on these specific benchmarks, due to the decoding strategy. LAMBADA for example is about predicting the next token, based on the previous tokens. This is the specific task the decoder is trained to do. While encoders are also trained by masking and predicting the word, it is trained by having the option to usually be able to depend on future context. Therefore it might not be able to capture information from only prior tokens as effectively.

BLiMP is a metric which tests for a strong syntactical understanding. The decoder is trained to generate coherent sequences which makes seman-

Config	Shared layers	Dec 10k	Enc 10k	Dec 25k	Enc 25k
Baseline	0	8.3 (22)	0.00 (1374)	8.93 (21)	1.998 (823)
Baseline scaled	0	8.00 (22)	0.019 (2150)	7.51 (22)	2.74 (605)
Deep cycle rev	6	7.22 (23)	0.00 (1418)	7.65 (22)	2.562 (551)
Wide cycle rev	6	0.89 (38)	0.00 (2175)	8.95 (20)	0.0 (3544)
Wide cycle	8	6.8 (-)	0.00 (1923)	4.09 (28)	0.097 (1248)
Deep cycle rev	10	6.66 (24)	0.19 (1384)	7.14 (22)	1.087 (1069)
Wide cycle rev	10	6.79 (25)	0.19 (1709)	9.84 (19)	0.78 (973)
Wide deep cycle rev	12	6.58 (24)	0.175 (1244)	9.33 (20)	1.144 (983)

Table 6: Lamabada accuracy results and the respective model configuration and number of shared layers for both encoder and decoder models trained for 10,000 and 25,000 steps. Perplexity is shown in parenthesis.

tic sense. This requires it to learn both a syntactically and semantically strong understanding of the previous sequence. While encoders also learn syntactical representations through its masking, the primary objective might lean more towards semantic understanding.

From Table 6 we can observe that as the number of steps increases both the decoder and encoder models do better on the lamabada metrics. A notable observation is the decoder’s results on 25,000 steps, where multiple of the trained models outperforms the accuracy score compared to the baseline.

5.2 Potential improvements

One interesting finding is that for the baseline, there was not much difference for the results between 10,000 and 25,000 steps. However for most of the parameter sharing models the results seemed to improve. This indicates that this is a technique which might become more efficient for even more training steps. This is further strengthened by the fact that the baseline was the best decoder model for 10,000 steps, while for 25,000 steps it was outperformed by multiple models.

Even though the encoders results were very poor for both 10,000 and 25,000 steps, we can see most of the models did improve significantly, both in accuracy and in perplexity when trained for more steps. The baseline was even outperformed when trained for 25,000 steps, while it was significantly better for 10,000 steps. The best model for 25,000 steps ended up being the scaled up baseline without parameter sharing.

6 Conclusion

In this paper we have built on the encoder decoder architecture from (Charpentier, 2024) by introducing layer-wise parameter sharing as in (Takase and

Kiyono, 2023). Our approach was to introduce parameter sharing, and compensate for the lost trainable parameters by scaling the model up to its original size.

Overall the experiments shows promising results for parameter sharing for some configurations within a decoder model, but this is not always the case. We can see a trend that by increasing the number of steps the parameter sharing models did improve more than the baseline models. However, we did not train for enough steps to see if they actually exceed the baseline, or only eclipses them.

By introducing parameter sharing we noticed a drop in results, and a time increase for the deep models. However by introducing 12 sharing layers in the decoder the parameter count was almost cut in half while the BLiMP only got reduced by around 3 points, as shown in Table 4. This indicates that while introducing parameter sharing in itself degrades the quality, the parameter decrease seems more significant. Therefore parameter sharing might be a better technique to downscale a well performing larger model, then to upscale smaller models in hope of better results.

References

- Lucas Charpentier. 2024. [Pretraining of tiny language models](#). (Accessed on 05/14/2024).
- BNC Consortium et al. 2007. British national corpus. *Oxford Text Archive Core Collection*.
- Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.
- 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*.

- Martin Gerlach and Francesc Font-Clos. 2020. A standardized project gutenberg corpus for statistical analysis of natural language and quantitative linguistics. *Entropy*, 22(1):126.
- John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1, ICASSP'92*, page 517–520, USA. IEEE Computer Society.
- Thamme Gowda and Jonathan May. 2020. [Finding the optimal vocabulary size for neural machine translation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3955–3964, Online. Association for Computational Linguistics.
- 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).
- Brian MacWhinney. 2000. The childes project. *Computational Linguistics*, 26(4):657–657.
- Richard Diehl Martinez, Hope McGovern, Zebulon Goriely, Christopher Davis, Andrew Caines, Paula Buttery, and Lisa Beinborn. 2023. [CLIMB – curriculum learning for infant-inspired model building](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 112–127, Singapore. Association for Computational Linguistics.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. [The LAMBADA dataset: Word prediction requiring a broad discourse context](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language models are unsupervised multitask learners](#).
- David Samuel, Andrey Kutuzov, Lilja Øvrelid, and Erik Velldal. 2023. [Trained on 100 million words and still in shape: BERT meets British National Corpus](#). In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1954–1974, Dubrovnik, Croatia. Association for Computational Linguistics.
- Sho Takase and Shun Kiyono. 2023. [Lessons on parameter sharing across layers in transformers](#). In *Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, pages 78–90, Toronto, Canada (Hybrid). 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](#). *CoRR*, abs/1706.03762.
- Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. [Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.
- Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. [BLiMP: The Benchmark of Linguistic Minimal Pairs for English](#). *Transactions of the Association for Computational Linguistics*, 8:377–392.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Tiny LM: Back to basics

Vebjørn Haug Kåsene
vebjorhk@uio.no

Peder August Darre Fasting
pafastin@uio.no

Abstract

We investigate the performance of encoder and decoder architectures with varying layer counts, aiming to identify their capabilities in understanding grammar and maintaining context. The models' success is evaluated using the BLiMP benchmark for grammar and the LAMBADA benchmark for contextual understanding.

Our results indicate that decoder models with deeper architectures enhance contextual understanding, while shallower decoders suffice for grammatical tasks. The best-performing models, a 6-layer decoder, and 12-layer decoder, showed competitive results compared to baseline models from the BabyLM challenge, despite their smaller parameter count and the small dataset size.

1 Introduction

The goal of this paper is to build on the research done in the BabyLM challenge (Warstadt et al., 2023a). The BabyLM challenge is a shared task that aims to find efficient methods for pretraining language models.

The challenge is inspired by the efficiency of human children learning language, only being exposed to about 100 million words before the age of 13 (Gilkerson et al., 2017). In contrast, state-of-the-art language models, like the LLaMa model (Touvron et al., 2023), need several orders-of-magnitude more data to be close to an average 13 year old level of language understanding.

The challenge itself is to pretrain a language model on a dataset that is approximately the same size as what an average 13 year old will have been exposed to. One of the main underlying hypothesis of the challenge is that being able to produce good architectures and methods for language modelling on a smaller scale may translate to better performance with larger models and larger datasets.

There are three different tracks that are given in the BabyLM challenge; the strict-small, strict and loose tracks. This paper is focused on the strict-small track where we are limited to training the models on an exclusive set of 10 million words, as opposed to the strict and loose tracks that allows for 100 million training words.

This paper has a single, self-imposed, limitation beyond the limitations already given in the challenge track; the parameter count of the models cannot exceed 7 million. Given the small dataset, we hypothesize that reducing the number of parameters is rational and that the main objectives of learning to identify context and grammar are possible at such a small scale.

Specifically the aim of this paper is to test encoder and decoder architectures separately, and experiment with the number of layers and their impact on how the models are able to understand grammar and context.

We measure the success of the models using two metrics, the BLiMP metric (Warstadt et al., 2020) with a primary focus on encoding grammar and the LAMBADA benchmark (Paperno et al., 2016) with a primary focus on keeping track of a larger context.

In addition to the two primary metrics, and in the spirit of the original challenge, we also focus on efficiency in training, timing the training duration for each candidate model.

2 Previous work

2.1 Papers submitted to the BabyLM challenge

The original BabyLM challenge received 31 papers (Warstadt et al., 2023b). Many different approaches were tried, curriculum learning, where training samples are sorted using different strategies to emulate a curriculum from easy to difficult (Oba et al., 2023; Hong et al., 2023; Mi, 2023), teacher student

knowledge distillation, which saw some success (Samuel, 2023; Timiryasov and Tastet, 2023), and formatting and splitting up the input data (Cheng et al., 2023). By far, the approach that saw the highest performance were the ones that experimented with the architectures and details thereof.

The overall winner of the original BabyLM challenge was the ELC-BERT model (Charpentier and Samuel, 2023), using optimizations on the transformer architecture from LTG-BERT (Samuel et al., 2023) and using layer weights. Charpentier and Samuel (2023) did not limit themselves on the model parameters. However, a few papers had this as an optional goal, hypothesizing that successful decisions made on smaller scale models are applicable in a larger setting.

2.2 Smaller models

A few of the submissions experimented with smaller models. Yang et al. (2023) experimented with a smaller version of a RoBERTa model (Liu et al., 2019) the BabyBERTa model (Huebner et al., 2021), further developing their own BabyBERTa+ model. The model was trained on the strict-small version of the BabyLM challenge dataset and achieved a competitive score on the BLiMP metric (Table 3). This result is in line with the findings of Zhang et al. (2021) that encoder-based models are good at capturing syntactic and semantic features.

Another encoder-based model that was submitted to the BabyLM challenge was the ELECTRA model (Fields et al., 2023), with a parameter count of only 7 million. However, the ELECTRA model was trained using a teacher-student based approach, which required a larger model to encode knowledge in the smaller model, consequently requiring the resources to run such a model.

Proskurina et al. (2023) experimented with both a small scale decoder and encoder, 66 million and 16 million parameters respectively. The encoder model (Bebeshka) performed poorly, on the BLiMP task, compared to other models using the same architecture, contrary to the previously mentioned findings of Zhang et al. (2021). While the decoder model (Zlata) performed worse than the baseline trained decoder, as can be seen in table 2 and table 3, even though the amount of parameters were far greater than the baseline decoder model.

Though, not a submission to the BabyLM challenge, Eldan and Li (2023) experimented with using small models (10 million parameters and less) on a dataset dubbed "tiny stories". The tiny stories

dataset used words that a 3 to 5 years old may understand, similar to the BabyLM challenge. They experimented with the number of layers and their corresponding sizes. They found that shallower models, i.e. fewer layers, are sufficient to learn grammar. Conversely, they found that generating language which is consistent with the start of the input text (context-tracking) requires deeper models.

2.3 Wide or deep?

The depth and width of a transformer architecture has also been researched before, though on a larger scale. Brown et al. (2022) and Xue et al. (2021) find that wider transformer networks may perform equally as well as deeper networks. At the same time they highlight one of the primary benefits of using wider networks being that one may be able to produce models with a smaller number of parameters and consequently the shallowness of such models may reduce training time by a large margin.

3 Hypotheses/research questions

Being informed by the previous research we pose the following questions: Are small scale decoders better at learning to interpret context than their encoder counterpart, given the same number of parameters? Given the findings of Zhang et al. (2021) are encoders well suited for learning grammar at a small scale, or are they in fact worse than their decoder counterpart, in line with the findings of Proskurina et al. (2023) and the baseline models (Charpentier, 2024)?

Furthermore, we hypothesize that grammar is a simple task for a transformer model to learn, consequently a model aimed at just learning grammar would require a fewer number of layers, in line with the findings of Eldan and Li (2023).

Given the task of keeping track of the context, we hypothesize that a greater number of layers may be required, however, instead of just adding more layers, we aim to illustrate where the benefits of more layers are diminished by experimenting on the depth of the models, while keeping the number of parameters approximately equal between the models.

In summation: our main research question is to find the relationship between the number of layers, the model architecture, the ability to keep track of context, and the ability to learn grammar.

4 Benchmarks

We compare our work with the benchmarks given by [Charpentier \(2024\)](#) (Table 1 and Table 2), and to a lesser degree with some models from the original BabyLM challenge ([Warstadt et al., 2023b](#)), specifically the ones focused on the strict-small track, and the models that use a smaller number of parameters (Table 3).

Model	Accuracy	Perplexity
Random Decoder	0.00	3578.65
Random Encoder	0.00	3611.27
Random Encoder-Decoder	0.00	3567.16
Trained Decoder	7.53	22.64
Trained Encoder	0.00	2349.43
Trained Encoder-Decoder	4.66	32.26

Table 1: Baseline LAMBADA scores ([Charpentier, 2024](#))

Model	Average
Random Decoder	51.56
Random Encoder	55.34
Random Encoder-Decoder	47.97
Trained Decoder	70.85
Trained Encoder	58.16
Trained Encoder-Decoder	58.13

Table 2: Baseline BLiMP scores ([Charpentier, 2024](#))

Model	Average
ELC-BERT (Charpentier and Samuel, 2023)	80
MLSM (Berend, 2023)	79
McGill-BERT (Cheng et al., 2023)	75
Bebeshka (16M encoder) (Proskurina et al., 2023)	55
Zlata (66M decoder) (Proskurina et al., 2023)	64.25
ELECTRA-tiny (7M) (Fields et al., 2023)	60 ¹
BabyBERTa+ (Yang et al., 2023)	69

Table 3: BLiMP scores from the strict-small version of the original BabyLM challenge ([Warstadt et al., 2023b](#))

¹Modified version of BLiMP

5 Data

The training data from the challenge is made up of different proportions of relatively simple language, designed to emulate what language a child of age 13 might have been exposed to ([Warstadt et al., 2023a](#)). We focus on the reduced dataset which consists of 10 million tokens compared to the full dataset that consists of 100 million tokens.

Most of the data is transcribed verbal language. Consequently, one could argue that this type of data does not necessarily lend itself well to learning strict grammatical rules, given the sometimes less than accurate grammar that can be found in transcribed text. For instance, part of our data, the CHILDES dataset ([Macwhinney, 2000](#)), is transcribed conversations between children and adults, which included sequences such as the following:

```
*CHI: Fraser.  
*MOT: Fraser.  
*MOT: uhhuh.  
*CHI: Cromer.  
*MOT: no.
```

For more details on the entire dataset, see the [Warstadt et al. \(2023a\)](#).

6 Method

6.1 Models

The models are based on the baseline models given by [Charpentier \(2024\)](#).

We train and test both pure encoder and pure decoder models with different numbers of layers (1, 2, 4, 6, 8, 12). We aim to maintain the size of the models at a similar level, with approximately 6 million parameters for each model. Each model is trained for 10000 steps with a batch size of 128.

To do this we change the number of parameters per block by changing the number of parameters in the attention layer and the feedforward layer. We aim to keep the ratio between the attention layer and feedforward layer approximately to a 2:1 ratio.

In total 12 models are trained and tested on the BLiMP and LAMBADA dataset. With the best scoring models on each benchmark trained 2 additional times with different seeds.

6.2 Performance metrics

This paper is focused on two zero-shot benchmarks: BLiMP ([Warstadt et al., 2020](#)) and LAMBADA ([Paperno et al., 2016](#)).

BLiMP is one of the primary metrics prescribed in the BabyLM challenge. It aims to measure the grammatical abilities of a language model. It does this by testing the language models ability to recognize the most grammatical sentence, of two sentences, as being the most likely. The metric is measured on a range of examples aimed at different grammatical concepts (Warstadt et al., 2020).

Commonly an issue with many architectures is forgetting important context, or being unable to use context in general. The LAMBADA dataset aims to test the amount of context a model is able to account for (Paperno et al., 2016). The LAMBADA dataset consists of examples of contexts, target sentences, and target words, where the target word is the last word of the target sentence. The aim of the model is then to guess the target word given the context and target sentence. In this paper we report the accuracy and perplexity of the models on the LAMBADA dataset targets.

In addition to these metrics we also time the training duration for each model, and report this as well.

7 Results

7.1 Training time

Architecture	Layers	training time
Decoder	12	21m 36s
	8	19m 21s
	6	16m 58s
	4	14m 41s
	2	12m 31s
	1	10m 32s
Encoder	12	24m 57s
	8	22m 33s
	6	20m 6s
	4	17m 57s
	2	15m 42s
	1	12m 10s

Table 4: Training times

Each model was trained on a single Nvidia RTX 3080 (10GB) over 10 000 steps. We recorded the training time for each model which are displayed in Table 4. The absolute training times were short across all models. This being one of the benefits to training smaller models allowing for more tuning and experimentation.

There is a pattern of shallower models being quicker to train, supporting the findings of Brown

et al. (2022). On a larger scale this pattern would likely be more extreme, again supporting shallower models.

7.2 BLiMP

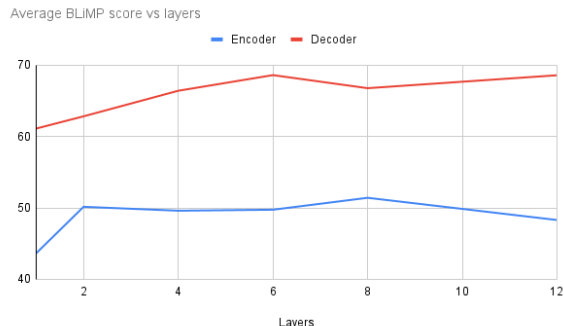


Figure 1: Average BLiMP score vs layers

Architecture	Layers	Average BLiMP
Decoder	12	68.58
	8	66.77
	6	68.60
	4	66.41
	2	62.83
	1	61.12
Encoder	12	48.31
	8	51.42
	6	49.75
	4	47.89
	2	51.33
	1	49.92

Table 5: Average BLiMP scores

Mean	Standard deviation
68.3	0.35

Table 6: Best model BLiMP score, 3 different seeds, decoder with 6 layers

In Table 5 we present the average BLiMP scores obtained by the encoder and decoder models with a different number of layers. Figure 1 shows a comparison between the encoder and decoder models average scores over number of layers. Overall we see that the decoder models perform better than all the encoder models, given the same number of layers. We also see that the number of layers becomes less important the more layers are added.

All the encoder models (Table 5) performed worse than the baseline models (Table 2). In fact,

the encoder models were worse than the randomly initialized encoder, indicating that the encoders did not learn anything meaningful.

We see that the decoder models performed well, compared to the encoder models. Compared to the baseline trained decoder model, the best performing decoder model performed slightly worse. However, the baseline model had exactly the same training and model parameters as our 12 layer model meaning that this discrepancy is likely due to a different batch size used to train our models.

The best performing model on the BLiMP dataset is the 6 layer decoder model (Table 6).

We see that our best performing model performs better than the Bebeska model (Proskurina et al., 2023), the Zlata model (Proskurina et al., 2023), and the ELECTRA-tiny model (Fields et al., 2023), despite all of them having more parameters. We also see that our best performing model performs about as well as the BabyBERTa+ model (Yang et al., 2023). (Table 3)

7.3 LAMBADA

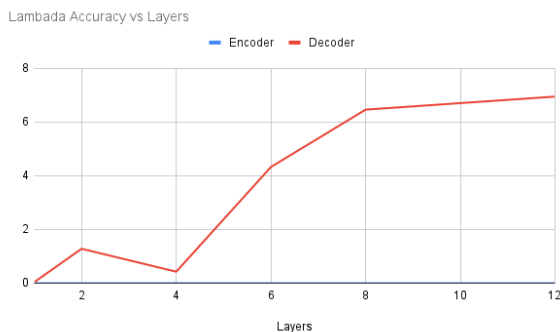


Figure 2: LAMBADA accuracy vs layers

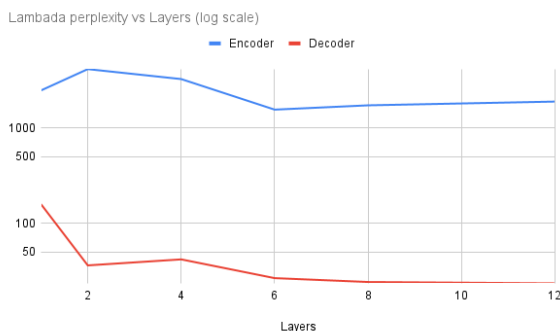


Figure 3: LAMBADA perplexity vs layers (log scale)

Architecture	Layers	Accuracy	Perplexity
Decoder	12	6.95	23.45
	8	6.46	24.23
	6	4.33	26.60
	4	0.43	41.84
	2	1.28	36.18
	1	0.04	158.48
Encoder	12	0	1892.29
	8	0	1728.40
	6	0	1557.91
	4	0	3259.10
	2	0	4144.76
	1	0	2478.21

Table 7: LAMBADA accuracy and perplexity for tested models

Metric	Mean	Standard deviation
Accuracy	6.08	0.75
Perplexity	24.40	0.835

Table 8: Best model LAMBADA score, 3 different seeds, decoder with 12 layers

In Table 7 we show the performance of our models on the LAMBADA dataset. Figure 2 and figure 3 shows a comparison of accuracy and perplexity, respectively, between the encoder and decoder models across different layers. As seen with the BLiMP scores, the decoder models emerges as the superior model on the LAMBADA task. The best performing model was the 12 layer decoder which still did not reach the same performance as the baseline decoder (Table 1). As discussed in the previous section, this is probably due to a different batch size or non deterministic initialization.

The encoder models performed similarly to the baseline encoder with a low accuracy and high perplexity. The scores being equally poor in both our experiments and the baseline may hint at problems elsewhere in the model and training parameters.

Table 8 displays the mean and standard deviation for the best performing model trained with 3 different seeds. The mean and standard deviation indicates that the seed used for the baseline and our models was on the higher end of the performance distribution for this model. Assessing the true impact of the additional 4 layers on the performance disparity between the 12-layer and 8-layer decoders remains ambiguous.

7.4 Discussion

The decoder models exhibit behaviors consistent with the observations of [Eldan and Li \(2023\)](#): more layers are required to understand context, whereas fewer layers may be sufficient for learning grammar. This pattern is evident in [Figure 2](#), where performance on the LAMBADA dataset improves significantly after four layers. However as evidenced by [Table 8](#) it is still unclear to what extent additional layers above 8 improve performance. In contrast, [Figure 1](#) shows that while the decoder model continues to improve up to six layers, the gains are not as pronounced as those observed with LAMBADA.

The encoder model scores ([Table 5](#)) may reveal that the BLiMP score does not improve significantly after the first layer. This may suggest that the models did not learn effectively, similar to the findings of [Proskurina et al. \(2023\)](#). We hypothesize that this may be due to the model architecture being more suited to decoder-only models, coupled with the limited vocabulary size. With a vocabulary of only 3072 tokens, the models may have faced challenges in making accurate predictions.

8 Conclusion

In this article we have experimented with different transformer model architectures on a small scale, specifically decoders and encoders, and the amount of layers in those architectures. We have mapped out the impact of the architectures and their layers on two tasks, grammar understanding and keeping track of context, through the BLiMP and LAMBADA metrics respectively.

Our findings show that the encoder model with our specific setting performed poorly compared to our expectation. While the decoder model performed better than many comparable models, more in line with the findings of [Proskurina et al. \(2023\)](#). We found that the decoder model only needed 6 layers to perform relatively well on the grammar task, compared to the benchmark models. We saw that increasing the amount of layers after this point, did not significantly boost the performance on the BLiMP task. This underpins the findings of [Eldan and Li \(2023\)](#), that fewer layers are required to understand grammar.

Furthermore, on the LAMBADA (context) task, the decoder model still performed better than its encoder counterpart. This was unsurprising, given the the decoder models architecture is specifically

designed to predict the last word given a sequence. Additionally, the model performed better the more layers we added, which is in line with the findings of [Eldan and Li \(2023\)](#).

9 Limitations and future work

For future work we recommend experimenting with different amounts of attention heads as well as looking at the impact of larger or smaller vocabulary sizes.

We did not experiment with how the different datasets, that compose the BabyLM dataset, impacted the scores. We theorize that this may have an impact on the scores.

While only having looked at the average score for BLiMP itself, future work could also analyze the impact of layers with regards to syntax, morphology and semantics.

The results of the BabyLM challenge showed that many different methods, not only relating to architecture, could have a significant impact on the scores, like curriculum learning. Combining some of the more successful approaches in the 2023 challenge would be an interesting direction to improve the best scores in the challenge even further.

In the spirit of the original challenge, it could be interesting to scale the best approaches to a larger parameter size and dataset setting, to see if the methods at a smaller scale are actually applicable to LLMs not just tiny LMs.

References

- Gábor Berend. 2023. [Better Together: Jointly Using Masked Latent Semantic Modeling and Masked Language Modeling for Sample Efficient Pre-training](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 298–307, Singapore. Association for Computational Linguistics.
- Jason Ross Brown, Yiren Zhao, Ilia Shumailov, and Robert D Mullins. 2022. [Wide Attention Is The Way Forward For Transformers?](#)
- Lucas Georges Gabriel Charpentier. 2024. [lgcharpe/IN5550_pretrain_tiny_models](#).
- Lucas Georges Gabriel Charpentier and David Samuel. 2023. [Not all layers are equally as important: Every Layer Counts BERT](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 238–252, Singapore. Association for Computational Linguistics.
- Ziling Cheng, Rahul Aralikkatte, Ian Porada, Cesare Spinoso-Di Piano, and Jackie CK Cheung. 2023.

- McGill BabyLM Shared Task Submission: The Effects of Data Formatting and Structural Biases. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 207–220, Singapore. Association for Computational Linguistics.
- Ronen Eldan and Yuanzhi Li. 2023. *TinyStories: How Small Can Language Models Be and Still Speak Coherent English?* *arXiv preprint*. ArXiv:2305.07759 [cs].
- Clayton Fields, Osama Natouf, Andrew McMains, Catherine Henry, and Casey Kennington. 2023. *Tiny Language Models Enriched with Multimodal Knowledge from Multiplex Networks*. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 47–57, Singapore. Association for Computational Linguistics.
- Jill Gilkerson, Jeffrey A. Richards, Steven F. Warren, Judith K. Montgomery, Charles R. Greenwood, D. Kimbrough Oller, John H. L. Hansen, and Terrance D. Paul. 2017. *Mapping the Early Language Environment Using All-Day Recordings and Automated Analysis*. *American Journal of Speech-Language Pathology*, 26(2):248–265.
- Xudong Hong, Sharid Loáiciga, and Asad Sayeed. 2023. *A surprisal oracle for active curriculum language modeling*. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 259–268, Singapore. Association for Computational Linguistics.
- Philip A. Huebner, Elior Sulem, Fisher Cynthia, and Dan Roth. 2021. *BabyBERTa: Learning More Grammar With Small-Scale Child-Directed Language*. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 624–646, Online. 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*.
- Brian Macwhinney. 2000. *The CHILDES project: tools for analyzing talk*. *Child Language Teaching and Therapy*, 8.
- Maggie Mi. 2023. *Mmi01 at The BabyLM Challenge: Linguistically Motivated Curriculum Learning for Pretraining in Low-Resource Settings*. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 269–278, Singapore. Association for Computational Linguistics.
- Miyu Oba, Akari Haga, Akiyo Fukatsu, and Yohei Oseki. 2023. *BabyLM Challenge: Curriculum learning based on sentence complexity approximating language acquisition*. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 290–297, Singapore. Association for Computational Linguistics.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. *The LAMBADA dataset: Word prediction requiring a broad discourse context*. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.
- Irina Proskurina, Guillaume Metzler, and Julien Velcin. 2023. *Mini Minds: Exploring Bebeshka and Zlata Baby Models*. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 30–40. ArXiv:2311.03216 [cs].
- David Samuel. 2023. *Mean BERTs make erratic language teachers: the effectiveness of latent bootstrapping in low-resource settings*. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 221–237, Singapore. Association for Computational Linguistics.
- David Samuel, Andrey Kutuzov, Lilja Øvrelid, and Erik Velldal. 2023. *Trained on 100 million words and still in shape: BERT meets British National Corpus*. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1954–1974, Dubrovnik, Croatia. Association for Computational Linguistics.
- Inar Timiryasov and Jean-Loup Tastet. 2023. *Baby Llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty*. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 279–289, Singapore. Association for Computational Linguistics.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. *LLaMA: Open and Efficient Foundation Language Models*. *arXiv preprint*. ArXiv:2302.13971 [cs].
- Alex Warstadt, Leshem Choshen, Aaron Mueller, Adina Williams, Ethan Wilcox, and Chengxu Zhuang. 2023a. *Call for Papers – The BabyLM Challenge: Sample-efficient pretraining on a developmentally plausible corpus*. *arXiv preprint*. ArXiv:2301.11796 [cs].
- Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023b. *Findings of the BabyLM Challenge: Sample-Efficient Pretraining on Developmentally Plausible Corpora*. In *Proceedings of the BabyLM Challenge at the 27th Conference on*

Computational Natural Language Learning, pages 1–34, Singapore. Association for Computational Linguistics.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohanney, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. **BLiMP: The Benchmark of Linguistic Minimal Pairs for English**. *Transactions of the Association for Computational Linguistics*, 8:377–392. Place: Cambridge, MA Publisher: MIT Press.

Fuzhao Xue, Ziji Shi, Futao Wei, Yuxuan Lou, Yong Liu, and Yang You. 2021. **Go Wider Instead of Deeper**. *arXiv preprint*. ArXiv:2107.11817 [cs].

Yahan Yang, Elicor Sulem, Insup Lee, and Dan Roth. 2023. **Penn & BGU BabyBERTa+ for Strict-Small BabyLM Challenge**. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 86–88, Singapore. Association for Computational Linguistics.

Yian Zhang, Alex Warstadt, Xiaocheng Li, and Samuel R. Bowman. 2021. **When Do You Need Billions of Words of Pretraining 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 1112–1125, Online. Association for Computational Linguistics.

A Repository

The repository for our code can be found here:
https://github.uio.no/pafastin/IN5550_pretrain_tiny_models.

(For security reasons, you cannot change the visibility of a fork.)

Tiny-LMs Project 2024: Curriculum Learning in Tiny Language Models

Magnus S. Dhelie

Institute for Informatics / University of Oslo

magnusdh@uio.no

Abstract

The development of large language models is resource-intensive and time-consuming, often requiring substantial computational power and expense. This project explores the potential of curriculum learning as a method to increase the efficiency of pre-training smaller, but better language models. This study aims to identify any significant improvements in learning performance that can be achieved through curriculum learning. The research involved training models on six datasets, and evaluating the performance using BLiMP and LAMBADA metrics.

The results show that curriculum learning does not improve the results of tiny language models. These results provide insights into the benefits and limitations of curriculum learning for tiny language models, offering a foundation for future exploration and potential scalability to larger models.

1 Introduction

Finding a pre-training method for large language models which improves efficiency is a task which could significantly reduce the cost and time of said training. However, using experimental methods to train large language models would likely just result in redundant use of resources where some methods could yield no improvements. Therefore, training tiny language models with different experimental methods and analyzing the results before scaling them up is a preferred way to test experimental learning methods.

A learning method which can yield improved results is performing curriculum learning. (Soviany et al., 2021) (Bengio et al., 2009) Attempting to mimic human learning from a young age, curriculum learning employs a gradual increase in complexity of data.(Elman, 1993)

This project attempts to address the following question: Does using curriculum learning improve

the performance of a tiny language model? Increased performance for a model will be defined as a better score in the evaluation metrics compared to the baseline models.

The limitations of the project are inspired by the BabyLM challenge of 2023(Warstadt et al., 2023), but in this case our models will be limited to a max of 7 million parameters rather than limiting the amount of words.

2 Curriculum Learning

Originally introduced in the paper "*Curriculum learning*" (Bengio et al., 2009), curriculum learning was proposed as a method where a gradual increase in complexity of the data was used in the training method. The idea behind using this gradual learning method was first introduced by (Elman, 1993) aiming to mimic the learning process of humans and animals. The first experiments performed by Elman (1993) only tested learning simple grammar with a recurrent network. However, the results reinforced that successful learning was a product of gradually increasing the complexity as the learning progressed.

The decision to choose a "curriculum learning-based" approach for training tiny language models is further supported by Bengio et al. (2009) where it is noted that "[...] simple multi-stage curriculum strategies give rise to improved generalization and faster convergence."¹ (Bengio et al., 2009) Thus, if implemented successfully for tiny language models it could potentially be scaled up and significantly decrease the amount of computational resources required for training even larger language models in a more effective manner alongside also performing better than models trained with a larger corpus.

This methodology is reinforced by the findings of Samuel et al. (2023) where the effects of down-scaling the training, and pre-training a model on

¹In the context of machine learning algorithms

a smaller, but carefully curated corpus showed results which outperformed the original BERT model. (Samuel et al., 2023)

3 Data

The datasets which have been used as vocabulary and training data in this project are: British National Corpus (2007) (BNC Consortium, 2007), CHILDES (MacWhinney, 2000), Project Gutenberg (Gerlach and Font-Clos, 2020), OpenSubtitles (Lison and Tiedemann, 2016), Simple English Wikipedia corpus (Wikimedia), and Switchboard (Godfrey et al., 1992)(Stolcke et al., 2000).

A simple overview can be used as a reference in Table 1 with the weight of each dataset also noted.

3.1 Calculating Perplexity

To make curriculum learning an effective method there needs to be a defined difficulty for the training data. Curriculum learning wants to be performed in a way which trains a model from easy to difficult tasks. This mimics the way humans learn as explained in Elman (1993).

The defined difficulty chosen for this project is the dataset perplexity. Perplexity is defined as the exponentiated average log-likelihood of a sequence. (HFP, 2023)

In simpler terms this definition means how well a language model predicts a sequence of words. Lower perplexity indicates that the model is more accurate in its predictions, while a higher perplexity indicates uncertainty in predictions and inaccurate guesses.

The formula for calculating perplexity is given as:

$$\text{PPL}(X) = \exp \left\{ -\frac{1}{t} \sum_{i=1}^t \log p_{\theta}(x_i | x_{<i}) \right\}$$

In this formula $\log p_{\theta}(x_i | x_{<i})$ is the probability assigned by the model to the word x_i given the previous words $x_{<i}$. In other words, when the model assigns a probability to each word in a given sequence based on the previous words, the logarithms of these probabilities are summed and averaged. The exponent of the negative of this average is the perplexity score for the entire sequence.

A simple explanation which follows the implementation for this study is that perplexity is calculated as the exponential of the average loss of a sequence.

There are two methods for calculating perplexity for datasets presented in Martinez et al. (2023). The first would be to calculate the perplexity of each dataset instance once at the start of training, while the other is to evaluate the perplexity of the remaining data at regular intervals.

For this project, the choice was made to evaluate each dataset once with a baseline model and sort them accordingly before training. This method allows the training to stay within a reasonable amount of training steps and overall training time, as it was noted in the project description to train in a reasonable time.

Calculating the perplexity of each dataset in its entirety proved time-consuming. Therefore we resorted to taking a percentage of samples from each dataset and calculating the perplexity for the dataset based on the samples. A sample size of 16% of each dataset was chosen and then combined them one dataset to create a tokenizer. The idea was to get a perplexity score which is representative of each dataset, but also not taking many hours to calculate. A downside to this sampling is that it leaves out a significant portion of each dataset. The datasets are also not of relatively equal size which could skew perplexity scores.

The perplexity for each dataset is then calculated using the baseline model for the architecture. The perplexity score from each dataset will be used to infer an order in which the training will occur.

The perplexity calculation can be seen in Table 2.

3.2 Implementing a curriculum

There are different ways one can implement a curriculum for a language model. The approach used for this implementation is an ordered approach, inspired by the CLIMB paper (Martinez et al., 2023). Having an order to the training instances can provide an increased effectiveness on the performance of a model during training (Schluter and Varab, 2018).

Following the results gained from the calculation, an ordered curriculum is decided for the model to train on. Each dataset will be trained on in its entirety in an ordered list. As curriculum learning starts with easier data and gradually increases to the more difficult data, the order of the training data based on Table 2 would be:

- Switchboard
- CHILDES

Corpus source	Weight	Domain	Citation
BNC 2007	8%	Dialogue	(BNC Consortium, 2007)
CHILDES	29%	Dialogue, Child-directed	(MacWhinney, 2000)
Project Gutenberg	26%	Fiction, nonfiction	(Gerlach and Font-Clos, 2020)
OpenSubtitles	20%	Dialogue, scripted	(Lison and Tiedemann, 2016)
Simple English Wikipedia	15%	Nonfiction	(Wikimedia)
Switchboard	1%	Dialogue	(Godfrey et al., 1992)(Stolcke et al., 2000)

Table 1: Datasets with their corresponding weight, domain, and citation.

The weight in percent per dataset indicates the size of the dataset when combined with the rest into one large dataset.

Corpus 16% size	Perplexity (bs=32)	Perplexity (bs=8)
BNC 2007	2320.05	2310.80
CHILDES	2093.17	2100.00
Project Gutenberg	2698.81	2702.69
OpenSubtitles	2184.08	2223.52
Simple English Wikipedia	3950.29	3957.58
Switchboard	1763.13	1816.97

Table 2: Datasets with their calculated perplexity for batch sizes (bs) 32 and 8 (not sorted)

The model used to calculate the perplexity is the baseline for the architecture, in this case an Encoder.

- OpenSubtitles
- BNC 2007
- Project Gutenberg
- Simple English Wikipedia

3.3 Pacing

After the difficulty is calculated and the curriculum has been determined, a pacing function needs to be set (Wu et al., 2020). The pacing function determines the pace at which the training will progress through the datasets.

Two primary functions which can be used in curriculum learning are a linear pacing or a logarithmic pacing, each providing their own benefits. For the linear pacing, a gradual increase in difficulty over time is expected. On the other hand, a logarithmic pace provides an introduction to easier concepts with diminishing increments based on the assumption that the model has learned the easier concepts. These functions have both been proposed in earlier literature published on curriculum learning (Bai et al., 2022) (Li et al., 2021).

The pacing function which we chose for the models employed a linear pacing function. The reason behind this choice is to simplify the process of doing curriculum learning where the model has to progress the same amount of steps for each dataset during training. For example: for a total of

6 datasets over 12000 steps, each dataset will be trained on for 2000 steps. ²

3.4 Baseline accuracies

Determining baseline accuracies is important as it provides scores which can be targeted for improvement. For this project the evaluation is performed using two methods: Benchmark of Linguistic Minimal Pairs (BLiMP) (Warstadt et al., 2020) and LAngeuage Modeling Broadened to Account for Discourse Aspect (LAMBADA) (Paperno et al., 2016).

Defined baselines are important as it shows the results we can expect from a certain baseline configuration. From this baseline attempts can be made to change hyper-parameters and training methods which may grant increased performance and better results.

Model	Accuracy	Perplexity
Baseline Encoder	0.00	1552.15
Baseline Decoder	9.47	20.99

Table 3: Baseline scores for the three architectures. Scored by LAMBADA.

²The function used for this is simply: $\text{Steps per dataset} = \frac{\text{Max steps}}{\text{Num of datasets}}$

Model	Average Accuracy
Baseline Encoder	45.21
Baseline Decoder	54.14

Table 4: Baseline scores for the three architectures. Scored by BLiMP.

4 Results

The results obtained from the models vary significantly depending on the hyper-parameters which are provided during training. To keep this simple and show a representative result for each architecture with the curriculum methods outlined, the model architectures are trained with identical hyper-parameters. This may not lead to optimal results for the specific architectures, but it is meant to show the performance differences and correlate them against one another.

For example, we want to know if the Encoder or Decoder architecture benefits more from one approach compared to another. Correlating certain hyper-parameter choices with increased performance can be beneficial for deciding further research and experimentation.

Throughout this section when models are referred to, models which are just named with their architecture name (e.g. "Decoder" or "Decoder 7M") are trained without curriculum learning using the standard combined dataset. Models trained using curriculum learning are referred to as "Curriculum" + the architecture name (e.g. "Curriculum Decoder").

4.1 7M parameter baselines

To stay within the limitation of 7M parameters set in this project a regular Encoder and Decoder model was trained with the configuration seen in Table 6.

This results in a total parameter count of 6,992,640 spread like this:

Architecture	Embedding	Classifier	Total
6,349,824	602,688	40,128	6,992,640

Table 5: Parameter count table for 7M parameter configuration.

Based on the scores from the evaluation in Table 6, a slight increase in accuracy can be noted when increasing the parameters to the set limit.

There are likely other configurations which could significantly increase the performance.

4.2 Curriculum Encoders

The Encoder architecture already performed worse than the Decoder baseline, but experimenting with a curriculum for this architecture could potentially still uncover if there were potential improvements based on the training method.

All the Encoder models were trained with a starting learning rate of $3 * 10^{-3}$ (0.003) and a final ratio of 0.1, indicating it scaled down to $3 * 10^{-4}$ (0.0003) over the course of the training. The scheduler which was used was a cosine method as this provides a steady decrease in learning rate to improve generalization and prevent overfitting.

However, for the curriculum learning method it was important that the model learned a significant amount from each dataset. Due to this, the learning rate had to be reset when a new dataset was introduced to the training.

Figure 1 shows the learning rate for the Encoder models. The long steady cosine is the learning rate for the baseline and 7M models trained without curriculum learning, while the repeating reset represent the learning rate for the models trained using curriculum learning.

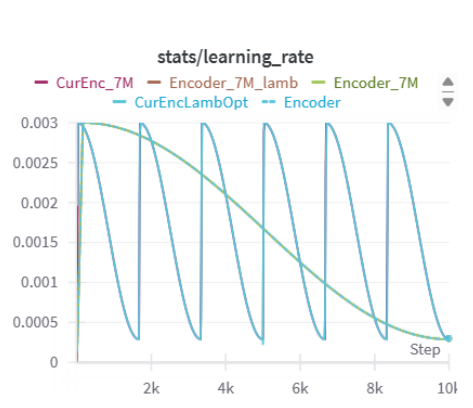


Figure 1: Encoder model learning rates. Data tracked using Weights and Biases³.

Comparing the statistics tracked during learning is quite difficult as the stats will be tracked for each dataset during the training. In Figure 2 we can observe that loss, perplexity, and accuracy are all quite poor compared to the models trained with 7M parameters on the baseline dataset containing all the datasets combined.

³<https://wandb.ai>

Model	Hidden size	Intermediate size	Attention heads	Hidden layers	Vocab size	BLiMP	LAMBADA
Encoder 7M	192	590	4	13	3072	45.96	0.00
Decoder 7M	192	590	4	13	3072	54.84	8.50

Table 6: Configuration for 7M parameter models with corresponding evaluation scores.

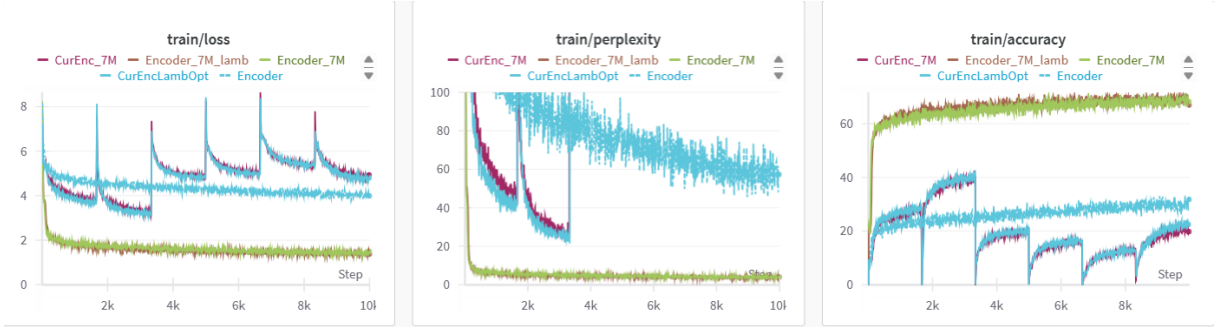


Figure 2: The loss, perplexity, and accuracy during training for Encoder models. Data is tracked using [Weights and Biases](#)³.

There may be multiple reasons for this, but the most likely reason is due to the fact that the model is using a dataset-specific tokenizer for each dataset. We experimented with using a single tokenizer combined on the entire dataset, but it proved difficult to implement in the learning process and settled for changing to a dataset-specific tokenizer for each dataset while training.

4.2.1 Encoder Model Performance Analysis

Table 7 shows a comparison of the baseline model, 7M parameter model, and the curriculum-training model using an Encoder architecture.

Model	BLiMP	LAMBADA (acc/perp)
Encoder (baseline)	45.21	0.00/1552.15
Encoder 7M	45.96	0.00/84579.93
Curriculum Encoder 7M	53.80	0.00/8001.26

Table 7: Encoder Models comparison

From the results of the evaluation, we observe that the curriculum-trained model performs better than the regular encoders on the BLiMP scoring. The average accuracy has increased which could indicate that the model has learned more generalized language patterns and understands language to a slightly higher degree.

We do however also observe that the average perplexity evaluated by LAMBADA is quite high. This shows that the model struggles with predicting the final word of a sequence, indicating that it does not have a good grasp on contextual understanding.

4.3 Curriculum Decoders

Already from the baseline results, it was clear that the Decoder architecture would outperform the Encoder models when it comes to contextual understanding as they (actually) scored on the LAMBADA test, and outperformed the Encoder on the BLiMP test. (See Table 4)

4.3.1 Decoder Model Performance Analysis

Model	BLiMP	LAMBADA (acc/perp)
Decoder (baseline)	54.14	9.47/20.99
Decoder 7M	54.84	8.50/21.66
Decoder 7M (.2 dropout)	54.62	6.48/24.06
Curriculum Decoder 7M	53.61	0.27/8826.31
12k steps		
Curriculum Decoder 12k	52.79	0.23/10383.47
Curriculum Decoder 12k (lamb optimizer)	53.68	0.29/15305.19
Curriculum Decoder 12k (lamb w/.2 dropout)	54.29	1.38/3172.47
Curriculum Decoder 12k (lamb batch128 lr5e-3)	53.85	1.20/3053.93

Table 8: Decoder Models comparison

In Table 8 we can see the results of the Decoder models compared with the curriculum trained Decoder models. The baselines already scored quite well and increasing to a 7M parameter configuration slightly increased the performance when evaluated using BLiMP.

The models trained using curriculum learning did not perform significantly worse when evaluated using BLiMP, but also did not perform better than the baselines. Alongside this, the models seem to not have learned contextual understanding as the model performance evaluated using LAMBADA has seen quite a drastic decrease.

A potential reason for this is the amount of tokenizers the curriculum model uses during its training, as it uses a specific tokenizer for each dataset.

This could possibly be hampering its learning ability for contextual understanding.

4.4 Extra dataset for curriculum

One method which was attempted for better comparison and results was to add the full segmented dataset back into the training loop. This was to get a comparison for how the curriculum-trained models were performing when it came to perplexity and accuracy on the same dataset which the regular models were trained on.

The extra dataset was added to the list of datasets to train on. It was added to the list so that it was trained on as the final dataset.

Important to note here is that when looping into the final dataset, the same tokenizer used for the regular models is also used for the curriculum model. This means the tokenization is done identically for the curriculum trained models and the non-curriculum trained models during the final dataset. The results can be seen in Figure 3.

The stats show that after having trained on the individual datasets before moving to the combined dataset the models perform significantly better with respect to loss, perplexity, and accuracy.

However this performance increase is not reflected in any way when the model is evaluated using the evaluation scripts as seen in Table 9.

Model	BLiMP	LAMBADA (acc/perp)
Decoder (baseline)	54.14	9.47/20.99
Decoder 7M	54.84	8.50/21.66
Decoder 7M (.2 dropout)	54.62	6.48/24.06
Curriculum Decoder Extra w/.2 dropout	50.93	0.00/84579.93
Curriculum Decoder Extra w/.3 dropout	50.97	0.08/120141.84

Table 9: Decoder Models with Extra dataset comparison.

5 Discussion

5.1 Encoders

The results we see in the evaluation are varied. For the Encoder models, the curriculum-trained model performs better when evaluated with BLiMP. (See Table 7). This is an indication that the Encoder model has increased performance when trained using a curriculum learning method. The LAMBADA score is worse when it comes to perplexity, but this is likely due to the tokenization process being done in a less-than optimal way with multiple tokenizers.

5.2 Decoders

The Decoders do not seem to have benefited from the curriculum learning as can be seen in Table 8. The models have lower performance from both evaluation metrics.

Based on the LAMBADA scores, it seems the models are much more confused and only have minimal, if any, contextual understanding. However, we believe these results are also a product of the tokenization process.

One notable observation is the results in Figure 3 where the stats show that the curriculum learning seems to have improved the performance of the models when training on the same dataset as the regular models. This could indicate that the curriculum learning method increases the performance of the models.

5.3 Performance from curriculum learning

Overall, the evaluation results do not seem to support the claim that curriculum learning increases performance. We hypothesize that this is a product of the tokenization process.

However, the statistics during training do indicate that the model could be performing better on the combined dataset during training. This may be an indicator that the model is learning more from this approach.

6 Conclusion

There are a couple of key conclusions which we can draw from these results.

The first one is that curriculum learning as a method works to train models. It appears to be a viable training method and does seem to have benefits when looking at the Encoder results.

The evaluation results however do not reflect any increased performance from the curriculum learning for the Decoder models and rather seems to have made the performance worse.

The stated research question was if curriculum learning could be used to improve the performance of tiny language models and from the results gathered in this study, the performance is better for Encoder models, but worse for Decoder models.

Overall, this implementation of curriculum learning did not succeed in improving the performance of tiny language models.

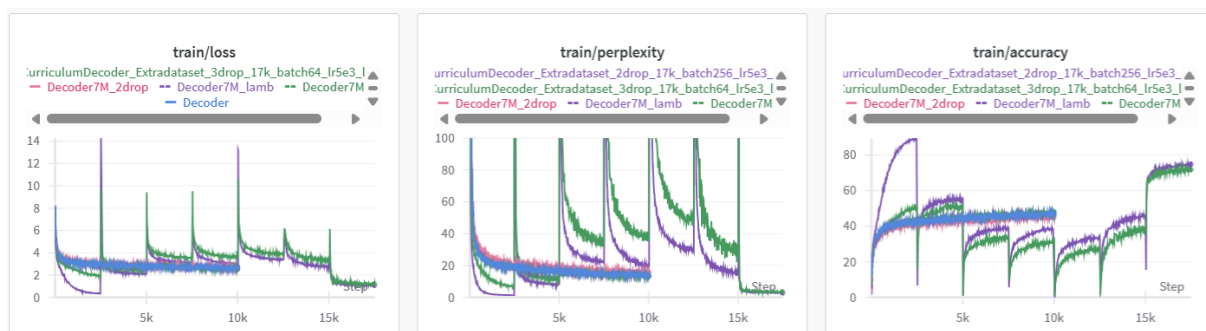


Figure 3: The loss, perplexity, and accuracy during training for Decoder models, and Decoder models with the extra dataset. Data tracked using [Weights and Biases](#)³.

Limitations

The scope of this project allowed for testing only a limited number of model configurations. Significant time was spent implementing curriculum learning which restricted the variety of developed models. Exploring a more broad range of configurations could identify more effective setups and improve baseline performance.

An idea for further research would be to improve the tokenization process which would likely yield better results for the curriculum-trained models. Performing the tokenization using a single tokenizer across all the datasets. This study did not manage to get that to work during the curriculum learning implementations, but it could likely lead to significantly better results as the model would not get confused from using multiple different tokenizers.

Changing up how the the curriculum learning approach may also yield interesting results. This implementation loops through each dataset for a given amount of steps, however one could also calculate the perplexity for each dataset and then order them correctly in a single dataset. That curriculum-ordered dataset could then be tokenized and used to train on. The results would potentially resolve the issues of multiple tokenizers.

Tracking statistics using WandB presented a challenge when attempting to compare the statistics of the baseline models to the curriculum models. The idea to add the combined dataset to the end of a curriculum training only emerged close to the end of the project. This did reveal interesting statistics which can be comparable, but the analysis is limited.

Finally, the model training time was limited and most models were trained for a total of 2 hours. Extending the training time to several hours or even

days could provide more comprehensive data and insights which are representative of a well-trained model.

References

- 2023. [Perplexity of fixed-length models — transformers 3.1.0 documentation](#). [Online; accessed 19. May 2024].
- He Bai, Tong Wang, Alessandro Sordoni, and Peng Shi. 2022. [Better Language Model with Hypernym Class Prediction](#). *ACL Anthology*, pages 1352–1362.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. [Curriculum learning](#). In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. Association for Computing Machinery, New York, NY, USA.
- BNC Consortium. 2007. [British National Corpus, XML edition](#). *University of Oxford*.
- J. L. Elman. 1993. [Learning and development in neural networks: the importance of starting small](#). *Cognition*, 48(1):71–99.
- Martin Gerlach and Francesc Font-Clos. 2020. [A Standardized Project Gutenberg Corpus for Statistical Analysis of Natural Language and Quantitative Linguistics](#). *Entropy (Basel)*, 22(1):126.
- J. Godfrey, E. Holliman, and J. McDaniel. 1992. [SWITCHBOARD: telephone speech corpus for research and development](#). *IEEE International Conference on Acoustics, Speech, and Signal Processing*.
- Conglong Li, Minjia Zhang, and Yuxiong He. 2021. [Curriculum Learning: A Regularization Method for Efficient and Stable Billion-Scale GPT Model Pre-Training](#). [Online; accessed 18. May 2024].
- Pierre Lison and Jörg Tiedemann. 2016. [OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles](#). *ACL Anthology*, pages 923–929.

- Brian MacWhinney. 2000. [The CHILDES Project: Tools for Analyzing Talk \(third edition\): Volume I: Transcription format and programs, Volume II: The database.](#) *Computational Linguistics*, 26(4):657.
- Richard Diehl Martinez, Hope McGovern, Zebulon Goriely, Christopher Davis, Andrew Caines, Paula Buttery, and Lisa Beinborn. 2023. [CLIMB – Curriculum Learning for Infant-inspired Model Building.](#) *ACL Anthology*, pages 112–127.
- Denis Paperno, Germán Kruszewski, Angeliki Lazariidou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. [The LAMBADA dataset: Word prediction requiring a broad discourse context.](#) *ACL Anthology*, pages 1525–1534.
- David Samuel, Andrey Kutuzov, Lilja Øvrelid, and Erik Velldal. 2023. [Trained on 100 million words and still in shape: BERT meets British National Corpus.](#) *ACL Anthology*, pages 1954–1974.
- Natalie Schluter and Daniel Varab. 2018. [When data permutations are pathological: the case of neural natural language inference.](#) *ACL Anthology*, pages 4935–4939.
- Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2021. [Curriculum Learning: A Survey.](#) *arXiv*.
- Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. [Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech.](#) *Computational Linguistics*, 26(3):339–373.
- Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell, editors. 2023. [Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning.](#) Association for Computational Linguistics, Singapore.
- Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel Bowman. 2020. [BLiMP: The Benchmark of Linguistic Minimal Pairs for English.](#) *Transactions of the Association for Computational Linguistics*, 8:377–392.
- Wikimedia. [Simple english wikipedia.](#)
- Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. 2020. [When Do Curricula Work?](#) *arXiv*.

3 Fact-Checking with Graph Evidence track

Enhancing Fact Verification with Hybrid Models: Integrating Graph Neural Networks and BERT Using the FactKG Dataset

Aleksandar Davidov

University of Oslo
aleksda@ifi.uio.no

Abstract

Abstract

This study explores the integration of Graph Neural Networks (GNNs) with BERT for the task of fact verification, utilizing the FactKG dataset. By incorporating subgraph evidence from DBpedia, the proposed Fact-checking model aims to enhance verification accuracy. The model achieved a validation accuracy of 92.59%, outperforming the BERT model that was fine-tuned on sentences alone, which achieved 91.52% accuracy. These results highlight the value of combining structured knowledge graphs with language models and GNNs, demonstrating that such hybrid approaches can significantly improve the accuracy of fact verification systems. The findings suggest that leveraging both structured and unstructured data allows for better handling of complex reasoning and relational information present in claims.

1 Introduction

The rapid diffusion of information in the digital age calls for effective fact-checking mechanisms to counter misinformation. Fact-checking, the practice of validating the truthfulness of claims using reliable evidence, has been recently put forward as one of the increased uses for fact-checking with structured data extracted from knowledge graphs, such as DBpedia (Kobilarov et al., 2009). Knowledge graphs open up a rich, interlinked structure of entities and relationships and have been used as a formalism encoding excellent, scalable representations of the web. Notably, Kim et al. (2023) presented the FactKG dataset, which collected over 108,000 claims, each associating with supporting or refuting sub-graphs from DBpedia. More specifically, very recent works by Wang et al. (2020) on path-based reasoning and Feng et al. (2020) on multi-hop relational reasoning have shown the importance of using KGs' relational paths for better

inference. More recently, the QA-GNN model Yasunaga et al. (2021) effectively integrates Graph Neural Networks (GNNs) and language models, demonstrating the power of unifying textual and graph data for complex reasoning tasks. This model highlights the potential for significantly advancing knowledge verification through joint reasoning over structured and unstructured data sources.

The central problem addressed is how to provide subgraph evidence from knowledge graphs in support of checking claims. Traditional text-based fact-checking systems lack the depth required to navigate the intricate relationship information in knowledge graphs. This work tries to enrich the existing fact-checking systems using subgraph evidence; we use BERT (Devlin et al., 2019) for processing the claim and GNNs based on multi-head attention to deal with subgraphs having relational data.

The primary objectives of this work are:

- Develop and implement an enhanced hybrid model combining BERT and GNNs
- Evaluate the performance of this model using the FactKG dataset
- Optimize the integration of subgraph evidence with text-based claims
- Verify the potential of combining BERT and GNNs to improve the precision of fact-checking systems

This research could have potentially have positive implications for enhancing automated fact-checking systems. By integrating sophisticated GNNs with BERT, this study aims to address the challenges associated with handling complex relational data. The outcomes of this research are directly relevant to news organizations, fact-checkers, and technology companies striving to maintain pub-

lic trust by improving the accuracy and reliability of fact-checking processes.

The scope of this research also includes assessing the performance improvement over conventional methods using the DBpedia subgraph. While the hybrid model shows promise, its weaknesses lie in the dependence on the quality and completeness of DBpedia and the computational complexity introduced with incorporating GNNs. These factors suggest directions for future exploration, aiming to further enhance the robustness and scalability of fact-checking systems.

2 Theory

2.1 Knowledge Graphs

In the digital age, fact-checking has become one of the most important elements for validating information and counterbalancing the rapid spread of false information (López-Marcos and Vicente-Fernández, 2021; Graves, 2018; Thorne et al., 2018). The growth of social media and digital platforms has led to an increase in misinformation, highlighting the critical need for automated fact-checking systems that leverage Artificial Intelligence (AI) and Natural Language Processing (NLP) to quickly process and validate enormous data sources (Vallayil et al., 2023; Nakov et al., 2021). These systems play a major role in ensuring public trust by delivering accurate information, thereby dramatically reducing the adverse impacts of false information (Graves, 2018).

Knowledge Graphs can be defined as directed, labelled multi-relational graphs integrating semantics. KGs are helpful in organizing heterogeneous data resources for coherent knowledge presentation, offering rich context to understand relationships among the data, and to interpret sophisticated queries that go beyond simple keyword searching (Kejriwal, 2022). One of the most common applications of graph-based knowledge is structured data organization, representing entities and their interrelations in a way that allows data integration and data retrieval. Besides that, they are a foundation for the intense application of AI, and through the storage of structured knowledge, AI systems acquire precise predictions to present reasonable high performance in decision-making (Chaudhri et al., 2022). More than that, KGs are broadly applied in the support of fact-checking tasks, where the verification of some claim’s accuracy needs the comprehensive understanding and analysis of

interconnected data.

2.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a modern model in NLP, proposed by Devlin et al. in 2018 (Devlin et al., 2019). BERT uses a transformer model to pretrain deep bidirectional representations conditioned on both left and right contexts in all layers. The procedure randomly masks a certain part of input and predicts tokens that have been masked, thus learns relationships of context inside the text (Liu et al., 2019; Koroteev, 2021).

This property of BERT, that it creates rich contextual embeddings, yet simultaneously captures large amounts of linguistic as well as world knowledge, seems essential for a number of NLP tasks, including question-answering and language inference (Lan et al., 2019), thus making BERT an important model. This capability is particularly crucial for fact-checking, as accurate interpretation of language and context is necessary to determine the veracity of claims.

2.2.1 Model Architecture

BERT’s architecture is based on Transformer architecture (Vaswani et al.), which is a multi-layer bidirectional self-attention model. We identify that the following are the critical components of BERT:

- **Layers and Blocks:** BERT-Base has 12 layers (or Transformer blocks), 768 hidden units, and 12 self-attention heads. At the same time, BERT-Large has 24 layers, 1024 hidden units.
- **Input Representation:** BERT accepts token embeddings as well as segment embeddings and position embeddings. The word embeddings that BERT uses are based on WordPiece embeddings with a vocabulary of size 30,000. In BERT, a special token for classification [CLS] is inserted at the beginning of each input. Segment separation within an input sequence is marked with [SEP] symbols.
- **Self-Attention Mechanism:** BERT uses self-attention mechanisms, which enables each token in a sentence to attend to every other potential token in that sentence; thus, it captures efficiently the notion of bidirectional context.

One important part of BERT architecture is the

self-attention mechanism, which can be shown as:

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

where Q (queries), K (keys), and V (values) are the input matrices, and d_k is the dimension of the keys (Vaswani et al.).

2.2.2 BERT for Binary Classification

BERT can be fine-tuned to binary classification tasks and output a binary prediction. The following are the steps involved during the process:

The loss function used for binary classification to train the model is the binary crossentropy, represented as:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (2)$$

where N is the number of training examples, y_i is the true label, and \hat{y}_i is the predicted probability.

2.3 Graph Neural Networks

GNNs is a class of neural networks that operates on graph data. A graph is an ordered pair $G = (A, \Sigma)$, where A is a set of edges and Σ is a set of nodes. GNNs are widely applied in such domains as social analysis, molecular biology, chemistry, and recommendation systems, since they can model complex dependencies and interactions (Wu et al., 2020; Hamilton et al., 2017).

2.3.1 Message Passing Mechanism

GNNs learn representations for nodes or the graph by propagating the information in the graph’s nodes and edges. Under the most basic mechanism, a node aggregates information from its neighboring nodes to update or revise its own local representation through a process known as message-passing. This is shown in the equation below:

$$h_i^{(k+1)} = \sigma \left(W^{(k)} \cdot \sum_{j \in \mathcal{N}(i)} h_j^{(k)} + b^{(k)} \right) \quad (3)$$

where $h_i^{(k)}$ is a feature vector of node i at layer k , $\mathcal{N}(i)$ are the neighbors of node i , and $W^{(k)}$ and $b^{(k)}$ are the learnable weights and biases at each layer with a non-linear activation function σ .

This is an iterative process through which nodes sequentially gather and integrate information from their neighborhoods to capture the structural and feature information in a graph (Kipf and Welling, 2017; Scarselli et al., 2009; Gilmer et al., 2017).

2.3.2 Graph Attention Networks

Graph Attention Networks (GATs) improved message passing by introducing attention coefficients, allowing nodes to weigh in on the importance of their neighbors’ features. The formulation of message passing in the GATs is expressed below:

$$h_i^{(k+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(k)} W^{(k)} h_j^{(k)} \right) \quad (4)$$

where $\alpha_{ij}^{(k)}$ are the attention coefficients calculated as:

$$\alpha_{ij}^{(k)} = \frac{\exp(\text{LR}(a^T [W^{(k)} h_i^{(k)} \| W^{(k)} h_j^{(k)}]))}{\sum_{k \in \mathcal{N}(i)} \exp(\text{LR}(a^T [W^{(k)} h_i^{(k)} \| W^{(k)} h_k^{(k)}]))} \quad (5)$$

Here, a is a learnable weight vector, and $\|$ denotes concatenation and LR is the LeakyReLU activation function. The attention mechanism enables GATs to differentiate the significance of neighboring nodes’ features, which can enhance the model’s ability to perform on various tasks (Velickovic et al., 2018).

3 Methods

3.1 Graphical model

In this work, our method adopts the GNN model of Brody et al. (2021), harnessing more expressible and stable GATv2 layers. This model is designed to efficiently propagate information across the nodes and edges of the graph in a way that updates node representations according to their local neighborhood. A key improvement in GATv2 is the implementation of dynamic attention, allowing the attention weights to vary based on the query node. This dynamic mechanism enhances the model’s ability to focus on different nodes depending on the context, thereby increasing its expressivity and flexibility. A very important part of the GNN model is its multi-head attention mechanism. At each GATv2 layer, multiple attention heads compute various attention coefficients, allowing the model to capture different aspects of the node features and

their neighbors. The feature aggregation in each layer is formulated as follows:

$$h_i^{(k+1)} = \sigma \left(\left\|_{m=1}^M \sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(k,m)} W^{(k,m)} h_j^{(k)} \right. \right) \quad (6)$$

where $h_i^{(k)}$ is the representation of node i at layer k , $\mathcal{N}(i)$ is the neighborhood of node i , $\alpha_{ij}^{(k,m)}$ is the attention coefficient computed by the m -th attention head at layer k , $W^{(k,m)}$ is the weight matrix of the m -th attention head at layer k , σ is a non-linear activation function, generally Leaky ReLU, and $\left\|_{m=1}^M$ denotes the concatenation of the outputs from all M attention heads.

Although the FactKG dataset does not explicitly utilize edge attributes, GATv2 aims to offer advantages in terms of enhanced expressiveness and stability. The dynamic attention mechanism allows the model to adjust the importance of nodes based on context, thus improving the model’s ability to capture complex relationships within the graph structure, even in the absence of edge features.

3.2 Fact-checking Model

The joint fact-checking model integrates outputs from a pretrained BERT model and a GNN to leverage both textual and subgraph data sources for fact-checking. This combined approach benefits from the strengths of each model: text-based learning with BERT and graph-based learning with GNN.

Input claims, in the form of textual data, are processed by the BERT model to develop high-dimensional embeddings. Specifically, the output representation of the BERT model for a given claim is denoted as \mathbf{t} , where $\mathbf{t} \in R^{d_t}$ and d_t represents the dimensionality of the BERT output.

Simultaneously, the GNN processes the knowledge graph associated with the claims, producing node embeddings. These embeddings are then aggregated to generate a graph-level representation. The graph-level representation \mathbf{g} is obtained after the final GNN layer by applying global mean pooling:

$$\mathbf{g} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} h_i^{(K)} \quad (7)$$

where $h^{(K)}$ represents the node embeddings from the last GNN layer and \mathcal{V} is the set of all nodes in the graph.

The textual representation \mathbf{t} from the BERT model and the graph representation \mathbf{g} from the

GNN are concatenated to form a single feature vector. This concatenation is given by:

$$\mathbf{h} = [\mathbf{t} \parallel \mathbf{g}] \quad (8)$$

where \parallel denotes the concatenation operator. Consequently, $\mathbf{h} \in R^{d_t+d_g}$, with d_g being the dimension of the GNN output. The final output y is computed as:

$$y = \sigma(\mathbf{W}\mathbf{h} + \mathbf{b}) \quad (9)$$

where \mathbf{W} is a weight matrix, \mathbf{b} is a bias vector, and σ is the sigmoid activation function, which is suitable for binary classification tasks.

3.3 FactKG dataset

For this work, we will be using utilizing the FactKG dataset, which is designed for fact verification via reasoning on knowledge graphs. It comprises of over 108 000 natural language claims, structured to facilitate reasoning through five distinct types: One-hop, Conjunction, Existence, Multi-hop, and Negation. Each claim is verified against DBpedia, a comprehensive and structured knowledge base, because FactKG leverages DBpedia’s vast, interconnected data to enhance the accuracy and transparency of the verification process. The dataset was created this way to utilize DBpedia’s logical structure and extensive information, which allows for complex reasoning patterns that simple datasets cannot support alone. The dataset features a mix of linguistic styles, including both colloquial and formal claims, broadening its applicability across different real-world scenarios each labeled as either supported or refuted based on the evidence provided by DBpedia.

Each claim is accompanied by several features, including:

- *Sentence*: The natural language claim to be verified. During data processing, the Sentence is concatenated with the paths found in the knowledge graph to form a comprehensive input for the model.
- *Label*: Indicates whether the claim is true (supported) or false (refuted) and is used during model training to supervise the learning process
- *Entity set*: The set of entities mentioned in the claim, serving as starting points for searching the knowledge graph. This evidence is

parsed and enhanced by searching the DBpedia knowledge graph to find additional connections. The Evidence is used to generate paths that connect entities, providing the necessary context to support or refute the claim.

- *Evidence*: The initial evidence supporting or refuting the claim, structured in a format that links entities through relationships. This evidence is parsed and enhanced by searching the DBpedia knowledge graph to find additional connections. The Evidence is used to generate paths that connect entities, providing the necessary context to support or refute the claim.
- *Metadata*: Additional information such as the claim style (written or colloquial) and reasoning type (e.g., one-hop, multi-hop, negation).

3.3.1 Utilization of DBpedia Knowledge Graph

The DBpedia knowledge graph is used to explore and extract meaningful paths that link the entities mentioned in each claim. This process begins with searching the graph for paths that connect the given entities through specified relationships. DBpedia facilitates uncovering both direct and indirect connections, which include multi-hop relationships that are not immediately apparent. These connections are crucial for comprehensive claim verification.

The process of finding relevant paths between entities in the knowledge graph involves several steps:

- *Initial Search*: Starting from the entities mentioned in the claim (*Entity set*), the search identifies potential relationships and paths within the knowledge graph.
- *Path Exploration*: The search explores various paths that connect these entities, considering both direct relationships and multi-hop connections that may provide additional context.
- *Path Categorization*: The paths are categorized into two types:
 - *Connected Paths*: Paths that directly link the entities. These paths provide clear, direct evidence supporting or refuting a claim.

- *Walkable Paths*: Paths that do not necessarily connect entities directly but still offer relevant information. These paths provide additional context and may reveal indirect connections that enrich the evidence set.

3.3.2 Data Processing and Transformation

Once the relevant paths are identified from DBpedia, they are processed and transformed into a structured format suitable for further analysis. This begins with parsing the evidence accompanying each claim to extract entities and their relationships. Lists of nodes (entities) and edges (relationships) are created, with each entity assigned a unique identifier and relationships represented as directed edges connecting the corresponding nodes. This step constructs mappings of entities to indices and enumerates the relationships between these entities. The evidence is then transformed into a graph representation suitable for input to a GNN.

4 Results

4.1 Training metrics

Below are the results achieved with the Fact-checking model, which consists of three GATv2 layers with 1024, 512, and 512 hidden units, respectively. A constant learning rate of $1e-05$ was used, and the model was trained on 4 epochs using the FactKG dataset. The BERT component of the model, which already has pre-trained weights, was fine-tuned during this process, while the GNN layers were trained from scratch.

Figure 1 shows the training loss curve of the Fact-checking model over batches.

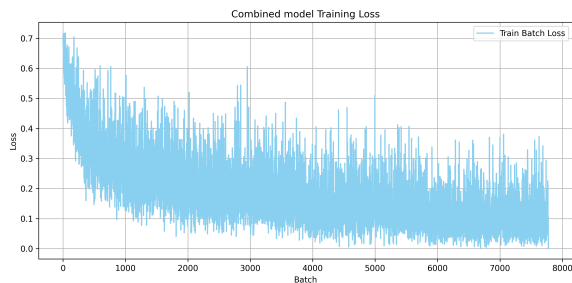


Figure 1: Training loss over epochs for the Fact-checking model.

As observed from the training curve, the loss steadily decreases with the number of epochs. However, there are noticeable fluctuations in the training process. These fluctuations could indicate some

instability in the training process or could be attributed to the model’s ability to handle different reasoning types effectively. The variation might be due to the complexity and diversity of the claims, including one-hop, multi-hop, and other reasoning types, which can introduce variability in the training dynamics.

Despite these fluctuations, there is a clear decreasing trend in the training loss over the number of batches. This consistent downward trend is a positive indication that the model is learning from the data, progressively improving its performance as it processes more training examples.

Further, figure 2 shows the validation batch losses over epochs.

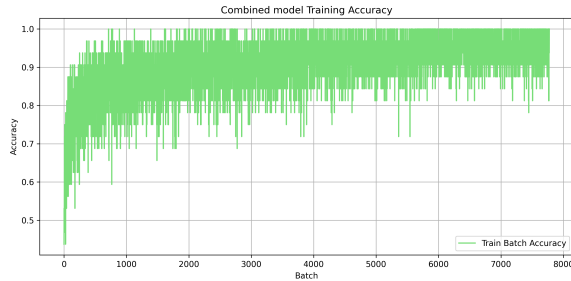


Figure 2: Training accuracy over epochs for the Fact-checking model.

Examining the accuracy curve for the Fact-checking model, it is evident that the model initially performs poorly, with accuracy below 50%, which suggests that it is essentially guessing at the beginning of the training process. However, the model quickly adapts to the data, and a steady increase in accuracy is observed as training progresses. Although there are fluctuations in the accuracy, these variations diminish over time, and an increasing number of batches achieve 100% accuracy. This upward trend in accuracy demonstrates the model’s improving capability to correctly verify claims as it learns from the training data.

4.2 Comparison

In this section, we compare the performance of the Fact-checking model against other models. Specifically, we evaluate:

1. A GNN with three GATv2 layers, having 1024, 512, and 512 hidden units respectively, trained with a constant learning rate of $1e-5$, using only subgraph evidence as input..
2. A BERT model fine-tuned to predict the label

of a claim based solely on the sentence (input natural language).

3. A BERT model that also incorporates subgraph evidence as input.

Both BERT models were fine-tuned with a learning rate of $1e-4$ with a learning rate schedule, and all models were trained for 4 epochs, as these parameters yielded the best results.

Table 1 provides us with the validation accuracies of the 4 different models.

Model	Val Accuracy (%)
Fact-checking Model	92.59
GNN with three GATv2 layers	77.31
BERT (sentence only)	91.52
BERT (with subgraph evidence)	92.93

Table 1: Validation accuracy comparison of the different models trained on 4 epochs.

The pure GNN implementation, which was trained solely on subgraph evidence, achieved a validation accuracy of 77.31%, which was the lowest among the evaluated models. This lower performance may be due to the GNN’s limited capacity to handle natural language data directly, relying solely on structured graph data. Additionally, the complexity of reasoning types, such as multi-hop and negation, might not be effectively captured by the GNN.

The BERT model fine-tuned on sentences alone achieved a validation accuracy of 91.52%. This demonstrates the effectiveness of pre-trained language models in understanding and classifying natural language data.

Incorporating subgraph evidence into the BERT model improved its performance, resulting in a validation accuracy of 92.93%. This indicates the benefit of using structured knowledge graph information to provide additional context and enhance claim verification.

The Fact-checking model, which integrates both the GNN and BERT components, achieved an accuracy of 92.59%. This performance suggests that combining graph-based reasoning with powerful language models adds value. Although the Fact-checking model did not surpass the BERT model with subgraph evidence, it outperformed the BERT model relying solely on sentences, indicating that the GNN contributes valuable relational insights that enhance overall model performance.

4.3 Future Work

Future work should aim to address the limitations identified in the current models and explore new directions to further improve fact verification. Specifically, further analysis is needed to determine the exact scenarios where the GNN performs well and where it falls short, particularly in relation to claim style and reasoning type. Understanding these aspects can provide insights into how different GNN architectures, such as comparing GATv2 layers with the original GAT layers, impact performance.

The promising results of incorporating subgraph evidence into the BERT model suggest that hybrid approaches should be further explored. Advanced methods for combining structured knowledge graph information with natural language processing, such as dynamically updating the knowledge graph during training or employing joint learning techniques, could provide more seamless integration and improved performance.

The promising results of incorporating subgraph evidence into the BERT model suggest that hybrid approaches should be further explored. Advanced methods for combining structured knowledge graph information with NLP, such as dynamically updating the knowledge graph during training or employing joint learning techniques, could provide more seamless integration and improved performance. Unlike utilizing a static knowledge graph, dynamically updating the graph during training could involve continuously refining and expanding the graph based on the model’s learning and feedback. This could include adding new relationships identified during training, adjusting the strengths of existing relationships, and incorporating temporal dynamics to reflect changes over time. These updates would allow the knowledge graph to evolve in real-time, and maybe providing more accurate and contextually rich representations that could enhance the model’s ability to verify claims.

For the Fact-checking model that integrates both GNN and BERT components, there is potential to refine the interaction between these components. Investigating different ways of combining these components, such as varying the interaction mechanisms or optimizing hyperparameters, could yield better results. Additionally, exploring multi-task learning approaches, where the model simultaneously learns to perform related tasks, could enhance its overall capability.

Furthermore, broader applicability and scalabil-

ity should be considered in future work. Evaluating the models on more diverse datasets, including different languages and domains, can help assess their generalizability.

Additionally, different reasoning types, such as multi-hop and negation, may pose significant challenges. Future research could focus on developing specialized modules or techniques to better handle these complex reasoning types. For example, integrating reinforcement learning approaches to navigate multi-hop paths more effectively or employing logic-based reasoning techniques which could potentially enhance the model’s capability to verify complex claims.

Lastly, comparing the proposed models against other state-of-the-art models, such as QA-GNN, could provide valuable benchmarks for evaluating effectiveness and highlighting areas for further improvement.

5 Discussion

The comparative analysis of the models provides valuable insights into the strengths and limitations of different approaches to fact verification. The pure GNN implementation, which was trained solely on subgraph evidence, achieved the lowest validation accuracy of 77.31%. This result highlights the challenge of relying exclusively on structured graph data to verify claims, as the GNN may struggle to capture the full semantic richness and contextual nuances of the claims without incorporating natural language data. The complexity of reasoning types, such as multi-hop and negation, may further exacerbate this limitation.

The BERT model fine-tuned on sentences alone demonstrated a substantial improvement, achieving a validation accuracy of 91.52%. BERT’s ability to leverage extensive contextual knowledge from large text corpora allows it to handle a variety of claim styles effectively, including both colloquial and formal language. However, there is room for improvement by enhancing the model’s ability to handle more complex reasoning types.

Incorporating subgraph evidence into the BERT model further enhanced its performance, achieving the highest validation accuracy of 92.93%. This demonstrates the added value of combining structured knowledge graph information with natural language processing. The additional context provided by DBpedia allows the model to understand and verify claims with complex reasoning patterns

more effectively. This improvement can be attributed to the richer and more structured information from the knowledge graph, which helps in capturing indirect relationships and multi-hop reasoning that the BERT model fine-tuned on sentences alone could not achieve.

The Fact-checking model, which integrates both GNN and BERT components, achieved a validation accuracy of 92.59%. Although it did not surpass the BERT model with subgraph evidence, it outperformed the BERT model using sentences alone. This indicates that the GNN contributes valuable relational insights that enhance overall model performance, particularly in understanding and verifying claims with complex relational structures. The slight underperformance compared to BERT with subgraph evidence suggests that while the integration of GNN adds value, there is still room for improvement in how the relational information is utilized.

6 Conclusion

In this study, we investigated the effectiveness of a Fact-checking model that integrates Graph Neural Networks (GNNs) with BERT for fact verification tasks. The model utilizes subgraph evidence from DBpedia to enhance the verification process. The results demonstrated that incorporating subgraph evidence through the GNN component improves the model’s performance compared to using BERT alone, highlighting the added value of leveraging structured knowledge graphs for relational insights.

The Fact-checking model, which combines GNNs and BERT, achieved a validation accuracy of 92.59%, showing that the GNN contributes valuable context and relational information that enhances the overall performance. However, the model did not surpass the BERT model with subgraph evidence, indicating that there is room for further refinement in how relational information is integrated.

The FactKG dataset showed to be a robust benchmark for evaluating these models, supporting various reasoning types and claim styles.

References

Shaked Brody, Uri Alon, and Eran Yahav. 2021. [How attentive are graph attention networks?](#) *arXiv preprint arXiv:2105.14491*.

Vinay K. Chaudhri, Chaitanya Baru, Naren Chittar, Xin Luna Dong, Michael Genesereth, James Hendler,

Aditya Kalyanpur, Douglas B. Lenat, Juan Sequeda, Denny Vrandečić, and Kuansan Wang. 2022. [Knowledge graphs: Introduction, history and, perspectives](#). *AI Magazine*, 43(1):17–29.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#). *arXiv preprint arXiv:1810.04805*.

Yang Feng, Xiang Chen, Bill Yuchen Lin, Po-Wei Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309. Association for Computational Linguistics.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. [Neural message passing for quantum chemistry](#). In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1263–1272.

Lucas Graves. 2018. [Understanding the promise and limits of automated fact-checking](#).

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30:1024–1034.

Mayank Kejriwal. 2022. [Knowledge graphs: A practical review of the research landscape](#). *Information*, 13(4).

Jinheon Kim, Seunghyun Park, Youngseo Kwon, Youngki Jo, James Thorne, and Edward Choi. 2023. [Factkg: Fact verification via reasoning on knowledge graphs](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16190–16206. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2017. [Semi-supervised classification with graph convolutional networks](#). *arXiv preprint arXiv:1609.02907*.

Georgi Kobilarov, Christian Bizer, Sören Auer, and Jens Lehmann. 2009. [Dbpedia—a linked data hub and data source for web applications and enterprises](#).

Mikhail Koroteev. 2021. [Bert: A review of applications in natural language processing and understanding](#). *arXiv preprint arXiv:2103.11943*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. [Albert: A lite bert for self-supervised learning of language representations](#). *arXiv preprint arXiv:1909.11942*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. [Linguistic knowledge and transferability of contextual representations](#). *arXiv preprint arXiv:1903.08855*.

- Casandra López-Marcos and Pilar Vicente-Fernández. 2021. [Fact checkers facing fake news and disinformation in the digital age: A comparative analysis between Spain and United Kingdom](#). *Publications*, 9(3).
- Preslav Nakov, David P. A. Corney, Maram Hasanain, Firoj Alam, Tamer Elsayed, Alberto Barrón-Cedeño, Paolo Papotti, Shaden Shaar, and Giovanni Da San Martino. 2021. [Automated fact-checking for assisting human fact-checkers](#). *CoRR*, abs/2103.07769.
- Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. [The graph neural network model](#). *IEEE Transactions on Neural Networks*, 20(1):61–80.
- James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. [FEVER: a large-scale dataset for fact extraction and VERification](#). 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 809–819, New Orleans, Louisiana. Association for Computational Linguistics.
- Manju Vallayil, Parma Nand, Wei Qi Yan, and Héctor Allende-Cid. 2023. [Explainability of automated fact verification systems: A comprehensive review](#). *Applied Sciences*, 13(23).
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. [Attention is all you need](#). *arXiv preprint arXiv:1706.03762*.
- Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. [Graph attention networks](#). *arXiv preprint arXiv:1710.10903*.
- Peifeng Wang, Nanyun Peng, Filip Ilievski, Pedro Szekely, and Xiang Ren. 2020. [Connecting the dots: A knowledgeable path generator for common-sense question answering](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4129–4140. Association for Computational Linguistics.
- Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2020. [A comprehensive survey on graph neural networks](#). *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24.
- Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. [Qa-gnn: Reasoning with language models and knowledge graphs for question answering](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546. Association for Computational Linguistics.

FactGenius: Combining Zero-Shot Prompting and Fuzzy Relation Mining to Improve Fact Verification with Knowledge Graphs

Sushant Gautam

Simula Metropolitan Center for Digital Engineering
Oslo, Norway
sushant@simula.no

Abstract

Fact-checking is a crucial natural language processing (NLP) task that verifies the truthfulness of claims by considering reliable evidence. Traditional methods are often limited by labour-intensive data curation and rule-based approaches. In this paper, we present FactGenius, a novel method that enhances fact-checking by combining zero-shot prompting of large language models (LLMs) with fuzzy text matching on knowledge graphs (KGs). Leveraging DBpedia, a structured linked data dataset derived from Wikipedia, FactGenius refines LLM-generated connections using similarity measures to ensure accuracy. The evaluation of FactGenius on the FactKG, a benchmark dataset for fact verification, demonstrates that it significantly outperforms existing baselines, particularly when fine-tuning RoBERTa as a classifier. The two-stage approach of filtering and validating connections proves crucial, achieving superior performance across various reasoning types and establishing FactGenius as a promising tool for robust fact-checking. The code and materials are available at <https://github.uio.no/sushantg/FactGenius>.

1 Introduction

Fact-checking is a critical task in natural language processing (NLP) that involves automatically verifying the truthfulness of a claim by considering evidence from reliable sources (Thorne et al., 2018). This task is essential for combating misinformation and ensuring the integrity of information in digital communication (Cotter et al., 2022). Traditional fact-checking methods rely heavily on manually curated datasets and rule-based approaches, which can be labour-intensive and limited in scope (Papadopoulos et al., 2024).

Recent advancements in large language models (LLMs) have shown promise in enhancing fact-checking capabilities (Choi and Ferrara, 2024). LLMs, with their extensive pre-training on diverse

textual data, possess a vast amount of embedded knowledge (Yang et al., 2024). However, their outputs can sometimes be erroneous or lacking in specificity, especially when dealing with complex reasoning patterns required for fact-checking. External knowledge, such as knowledge graphs (KGs) (Hogan et al., 2021), can aid in fact-checking.

In this paper, we propose FactGenius, a novel approach that combines zero-shot prompting of LLMs with fuzzy relation-mining technique to improve reasoning on knowledge graphs. Specifically, we leverage DBpedia (Lehmann et al., 2015), a structured source of linked data, to enhance the accuracy of fact-checking tasks.

Our methodology involves using the LLM to filter potential connections between entities in the KG, followed by refining these connections through Levenshtein distance-based fuzzy matching. This two-stage approach ensures that only valid and relevant connections are considered, thereby improving the accuracy of fact-checking.

We evaluate our method using the FactKG dataset (Kim et al., 2023b), which comprises 108,000 claims constructed through various reasoning patterns applied to facts from DBpedia. Our experiments demonstrate that FactGenius significantly outperforms existing baselines (Kim et al., 2023a), particularly when fine-tuning RoBERTa (Liu et al., 2019) as a classifier, achieving superior performance across different reasoning types.

In summary, the integration of LLMs with KGs and the application of fuzzy matching techniques represent a promising direction for advancing fact-checking methodologies. Our work contributes to this growing body of research by proposing a novel approach that effectively combines these elements, yielding significant improvements in fact-checking performance.

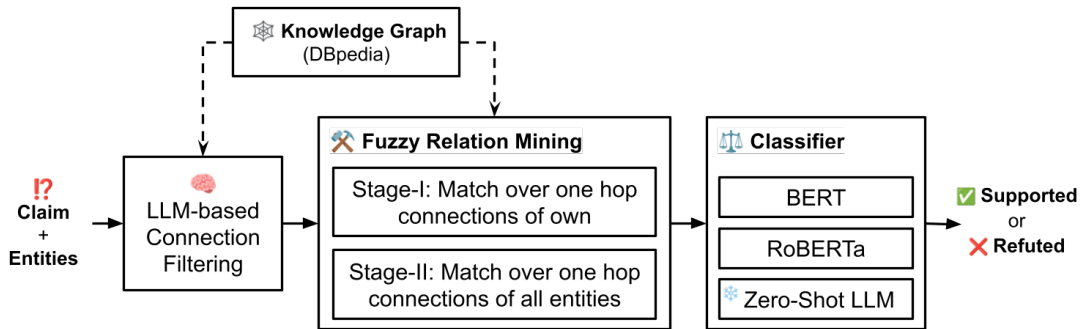


Figure 1: Overall pipeline of FactGenius: The process starts with LLM-based Connection Filtering using a knowledge graph (see Section 3.3.1). In Fuzzy Relation Mining (see Section 3.3.2), Stage-I matches one-hop connections of entities, and optionally, Stage-II includes all entities’ connections. The classifier (BERT, RoBERTa, or Zero-Shot LLM; see Section 3.4) then determines if the claim is supported or refuted.

2 Literature Review

Fact-checking has become an increasingly vital aspect of natural language processing (NLP) due to the proliferation of misinformation in digital communication (Guo et al., 2022). Traditional approaches to fact-checking have typically relied on manually curated datasets and rule-based methods, which, while effective in controlled environments, often struggle with scalability and adaptability to new types of misinformation (Saquete et al., 2020; Guo et al., 2022). The labour-intensive nature of these methods also poses significant challenges in rapidly evolving information landscapes (Nakov et al., 2021; Zeng et al., 2021).

To address challenges in understanding machine-readable concepts in text, FactKG introduces a new dataset for fact verification with claims, leveraging knowledge graphs, encompassing diverse reasoning types and linguistic patterns, aiming to enhance reliability and practicality in KG-based fact verification (Kim et al., 2023b). Similarly, the Fact Extraction and VERification (FEVER) dataset (Thorne et al., 2018) pairs claim with Wikipedia sentences that support or refute them, providing a benchmark for fact-checking models. The authors employed a combination of natural language inference models and information retrieval systems to assess claim veracity. The GEAR framework (Zhou et al., 2019) improves fact verification by using a graph-based method to aggregate and reason over multiple pieces of evidence, surpassing previous methods by enabling evidence to interact.

Recent advancements in large language models (LLMs) have demonstrated considerable potential in enhancing fact-checking processes (Kim et al., 2023a; Choi and Ferrara, 2024). LLMs have been pre-trained on vast and diverse corpora (Yang et al., 2024), enabling them to generate human-like text and possess a broad knowledge base (Choi and Ferrara, 2024). However, despite their impressive capabilities, LLMs can produce outputs that are erroneous or lack the specificity required for complex fact-checking tasks (Choi and Ferrara, 2024). This is particularly evident when intricate reasoning and contextual understanding are necessary to verify claims accurately (Chai et al., 2023). Several studies have explored the integration of LLMs with external knowledge sources to improve their performance in fact-checking tasks (Cui et al., 2023; Ding et al., 2023).

The incorporation of knowledge graphs (KGs) into fact-checking frameworks has also garnered attention. KGs, such as DBpedia (Lehmann et al., 2015), provide structured and linked data that can enhance the contextual understanding of LLMs.

Knowledge graphs have been used to improve various NLP tasks by providing additional context and relationships between entities, as demonstrated by initiatives for knowledge-aware language models (Li et al., 2023; Logan Iv et al., 2019) and KG-BERT (Yao et al., 2019).

Approximate string matching (Navarro, 2001), also called fuzzy string matching, is a technique used to identify partial matches between text strings. Fuzzy matching techniques (Navarro, 2001) have been applied to enhance the integration of LLMs and KGs (Wang et al., 2024).

Levenshtein distance-based similarity measure (Levenshtein et al., 1966) helps in identifying strings which have approximate matches which can be useful to find relevant connections between entities by accommodating minor discrepancies in data representation. This approach has been beneficial in refining the outputs of LLMs, ensuring that only valid and contextually appropriate connections are considered (Guo et al., 2023).

Our proposed method, FactGenius, builds on these advancements by combining zero-shot prompting of LLMs with a fuzzy relation-mining technique to improve reasoning over KGs. This methodology leverages DBpedia as a structured source of linked data to enhance fact-checking accuracy. By using LLMs to filter potential connections between entities and refining these connections through fuzzy matching, FactGenius aims to address the limitations of existing fact-checking models.

3 Methodology

FactGenius leverages the capabilities of a Large Language Model (LLM) to filter possible connections between entities in a Knowledge Graph (KG), particularly utilizing DBpedia (Lehmann et al., 2015) as a structured source of linked data.

Since the output of LLMs can be erroneous, the connections are further refined and enriched using Levenshtein distance (Levenshtein et al., 1966) and are also validated to ensure that such connections exist. This process is crucial for tasks such as fact-checking, where establishing valid and relevant connections between entities can validate or refute claims. Finally, the classifier, which can be fine-tuned over pre-trained models like BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019), or a Zero-Shot LLM, determines whether the claim is supported or refuted. The overall pipeline is shown in Figure 1.

3.1 Dataset

The FactKG dataset (Kim et al., 2023b) is used which comprises 108,000 claims constructed through various reasoning patterns applied to facts sourced from DBpedia (Lehmann et al., 2015). Each data point consists of a natural language claim in English, the set of DBpedia entities mentioned in the claim, and a binary label indicating the claim’s

veracity (Supported or Refuted). The distribution across labels and five different reasoning types is shown in Table 1. The relevant relation paths starting from each entity in the claim are known which aids in the evaluation and development of models for claim verification tasks.

The dataset is accompanied by processed DBpedia, an undirected knowledge graph (KG). The dataset provides researchers with a valuable resource for exploring the intersection of natural language understanding and knowledge graph reasoning.

Table 1: Data distribution across labels and five reasoning types.

Set	Train	Valid	Test
Total Rows	86367	13266	9041
True (Supported)	42723	6426	4398
False (Refuted)	43644	6840	4643
One-hop	15069	2547	1914
Conjunction	29711	4317	3069
Existence	7372	930	870
Multi Hop	21833	3555	1874
Negation	12382	1917	1314

3.2 Claim Only Classifier

In this setting, where the models are given only the claim and tasked with predicting the label, it is expected that the model will heavily depend on stored evidence within its trained weights or identify patterns within the structure of the provided claims.

3.2.1 Zero-shot Claim Only Baseline

A baseline is established using the Meta-Llama-3-8B-Instruct¹ (Meta, 2024) model with zero-shot prompting for claim verification, asking it to verify the claim without evidence. Through instruction prompt engineering, it is ensured that the model responds with either 'true' or 'false'. A retry mechanism is implemented to handle potential failures in LLM responses. A prompt example is shown in Figure 2.

3.2.2 RoBERTa as Claim Only Fact Classifier

RoBERTa-base² is fine-tuned with claims as input, training it to predict Supported or Refuted. This is to compare with the BERT baseline reported in previous works (Kim et al., 2023b).

¹huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

²huggingface.co/FacebookAI/roberta-base

```

[{"role": "system", "content":
  "You are an intelligent fact checker trained on Wikipedia.
  You are given a single claim and your task is to decide
  whether all the facts in the given claim are supported
  by the given evidence using your knowledge.
  Choose one of {True, False}, and output the one-sentence
  explanation for the choice. "
}, {"role": "user", "content":
  ""
}, {"role": "system", "content":
  ""
}, {"role": "user", "content":
  ""
}]

```

Figure 2: Example prompt given to Llama3-Instruct without evidence for zero-shot fact-checking. <<< ... >>> signs are added just to indicate that the content inside is different for each prompt.

```

[{"role": "system", "content":
  "You are an intelligent graph connection finder. You
  are given a single claim and connection options for the
  entities present in the claim. Your task is to filter
  the Connections options that could be relevant to connect
  given entities to fact-check Claim1. ~ ( tilde ) in the
  beginning means the reverse connection. "
}, {"role": "user", "content":
  ""
}, {"role": "system", "content":
  ""
}, {"role": "user", "content":
  ""
}]

```

Figure 3: Example prompt given to Llama3-Instruct to filter potential connections between entities based on a given claim.

3.3 Graph Filtering

The graph filtering is divided into two main stages:

3.3.1 Filtering Possible Connections

This stage involves utilizing an LLM, particularly the Llama3-Instruct model, to identify and filter potential connections between entities based on a

given claim. The detailed steps are as follows:

Data Preparation Entity sets and their possible connections are extracted from the KG (DBpedia). Each entity and its associated possible connections form the initial input for the LLM.

LLM Integration The LLM is tasked with identifying relevant connections for each entity in the specific claim. The process involves:

1. Encoding each entity and its possible connections into a structured format suitable for the LLM.
2. Utilizing the LLM's inference capabilities to filter out irrelevant connections based on the context provided by the claim.
3. Generating a filtered set of connections in a structured format, which is then evaluated for completeness and relevance.

An example of the prompt used with LLM in Stage-I is shown in Figure 3. Prompts are crafted through iterative testing and refinement, aiming to optimize results and performance.

Handling Invalid LLM Response A retry mechanism is implemented to handle potential failures in LLM responses. If the LLM output is inadequate (e.g., empty or nonsensical), the request is retried up to a specified maximum number of attempts, typically 10. Throughout this experiment, however, we did not encounter any cases where the retry exceeded this limit.

3.3.2 Fuzzy Relation Mining

The LLM-filtered connections are then validated against the KG to ensure their existence and relevance. This involves:

1. **Stage-I:** Checking each connection filtered using LLM against the KG to confirm its validity. For each connection in the entities, perform fuzzy matching using Levenshtein distance to match entities in the first-hop relation of the graph. This approach accommodates speckling and reverse connection errors.
2. **Stage-II:** Matching potential connections fuzzily, while considering reverse relationships and similarities across all the one-hop connections in the knowledge graph of all entities within the claims.

The details are explained in Algorithm 1.

Algorithm 1 Relationship Mining with Validation

```
1: Input:  $A$  - dictionary of entities with their connections,  $G$ : Graph
2: Output: probable_connections- dictionary of entities with
   updated and validated connections
3: procedure VALIDATERELATION( $A$ )
4:   Initialize: probable_connections: {}
5:   — Stage-I —
6:   for each entity, connections in  $A$  do
7:     Retrieve: all one_hop_connections for entity  $G$ 
8:     for each connection in connections do
9:       Fuzzily match from one_hop_connections
10:      Filter matches with a similarity score greater than 90
11:      Update entity in probable_connections
12:     end for
13:   end for
14:   — Stage-II (optional) —
15:   all_connections = all connections in probable_connections
16:   for each entity, connections in probable_connections do
17:     Retrieve: all one_hop_connections for entity  $G$ 
18:     for each connection in all_connections do
19:       Fuzzily match from one_hop_connections
20:       Filter matches with a similarity score greater than 90
21:       Update entity in probable_connections
22:     end for
23:   end for
24: end procedure
```

3.4 With Evidence Classifier

In this configuration, the model is supplied with both the claim and graphical evidence as input, and it then makes predictions regarding the label. FactGenius utilizes graph filtering, as explained in Section 3.3, to ensure retention of the most relevant and accurate connections.

3.5 Zero-shot LLM as Fact Classifier

This involves utilizing Llama-3-Instruct as a fact classifier, to predict Supported or Refuted for the given input claim and evidence. A retry mechanism is implemented to handle potential failures in LLM responses. A prompt example with evidence is shown in Figure 4.

3.6 Fine-tuning pre-trained models

Pre-trained BERT-base-uncased³ and RoBERTa-base are finetuned with claim and evidence as inputs to predict whether the claim is supported or refuted. In addition, an ablation evaluates the contributions of each stage of our approach. This involved sequentially removing Stage-II and measuring the performance of the system after the removal. The results of the ablation study

³huggingface.co/google-bert/bert-base-uncased

allowed us to quantify the impact of both stages on the overall performance of the model. Accuracy as an evaluation metric across all reasoning types was employed to quantify the performance improvements resulting from the ablation study.

```
[[
  "role": "system", "content":
  "You are an intelligent fact-checker. You are given a
  single claim and supporting evidence for the entities
  present in the claim, extracted from a knowledge graph.
  Your task is to decide whether all the facts in the given
  claim are supported by the given evidence.
  Choose one of {True, False}, and output the one-sentence
  explanation for the choice. "
], [
  "role": "user", "content":
  '''
  ## TASK:
  Now let's verify the Claim based on the evidence.
  Claim:
  << < Well, The celestial body known as 1097 Vicia has a
  mass of 4.1kg.> > >
  Evidences:
  << <
  1999_Hirayama >- mass -> ""4.1""
  1097_Vicia >- mass -> ""9.8""
  > > >
  #Answer Template:
  "True/False (single word answer),
  One-sentence evidence."
  ]]
```

Figure 4: Example prompt given to Llama3-Instruct with evidence for zero-shot fact-checking.

3.7 Implementation

Our FactGenius system implementation leverages several advanced tools and frameworks to ensure efficient and scalable processing. The Llama3-Instruct inference server is set up using vLLM (vLLM Project, 2024; Kwon et al., 2023), running on an NVIDIA A100 GPU (80 GB vRAM) to facilitate rapid inference. This server runs standalone, integrating seamlessly with the FactGenius pipeline.

For model fine-tuning and evaluation, we employ the Hugging Face Transformers library, utilizing the Trainer class for managing the training process. This setup allows for the fine-tuning of pre-trained models like BERT and RoBERTa on our pipeline. Hyper-parameters such as batch size, learning rate, and training epochs are configured to optimize performance, with computations accelerated by PyTorch.

4 Results

To evaluate the performance of our proposed methods, we conducted a series of experiments comparing different strategies for fact-checking. The results are summarized in Table 2.

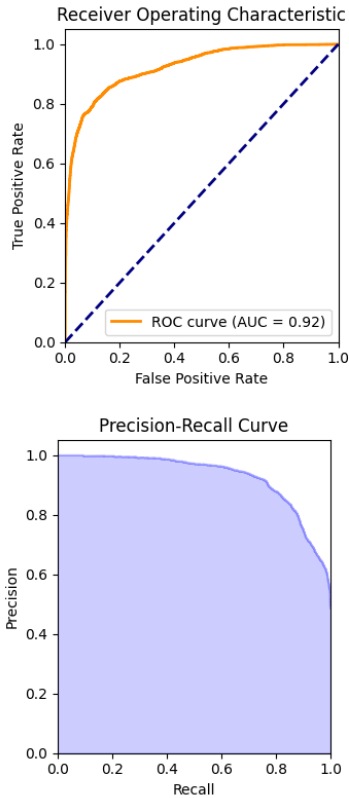


Figure 5: ROC curve (top) illustrates classifier discrimination ability with AUC 0.92, while Precision-Recall curve (bottom) reveals precision across recall levels for the best model in the test dataset.

4.1 Baseline and Claim Only Models

This achieved an accuracy of 0.61, demonstrating the inherent knowledge embedded within the Llama3 about the facts in our corpus. Adding evidence to the Llama3-Instruct model’s instructions significantly improved its accuracy from 0.61 to 0.68. This reflects the trivial phenomenon that, incorporating relevant evidence can enhance fact-checking performance in a zero-shot learning scenario where the performance is mostly dependent on knowledge embedded in the model.

4.2 Comparison of Different Models

We compared the performance of RoBERTa, under the claim-only scenario. RoBERTa outperformed the reported accuracy of BERT (Kim et al., 2023b),

achieving an accuracy of 0.68, which is on par with the 12-shot ChatGPT model reported in the KG-GPT paper (Kim et al., 2023a). This suggests that RoBERTa is a highly effective model for fact-checking tasks.

4.3 Incorporating Evidence with FactGenius

We employed a zero-shot prompting technique to filter relevant connections from the evidence, followed by fuzzy matching across multiple stages. This approach enabled us to retrieve evidence by searching over the graph. However, directly applying zero-shot prompting with Llama3-Instruct, even with evidence, did not yield superior performance. When using fine-tuned BERT as a classifier, the performance was comparable to the 12-shot KG-GPT model. However, fine-tuning RoBERTa led to a significant performance boost, achieving an accuracy of 0.85, the highest among all models tested, even surpassing the GEAR baseline (Zhou et al., 2019), which enhances fact verification by using a graph-based approach to aggregate and reason over multiple pieces of evidence.

4.4 Two-Stage Approach

To assess the contribution of our two-stage approach, we first apply only the first-stage graph filtering method (Stage-I) to filter the relationships, which achieved an accuracy of 0.83. Incorporating the second stage (Stage-II) further improved the performance to 0.85. The second stage particularly enhanced performance across all reasoning types, with notable improvements in conjunction and negation tasks. Interestingly, for the existence reasoning type, the BERT classifier performed on par with the best models, indicating its robustness for this specific task.

4.5 Overall Performance

FactGenius with a fine-tuned classifier, demonstrated superior performance across all reasoning types compared to previously reported accuracies. This validates the effectiveness of our multi-stage evidence retrieval and classifier fine-tuning approach in improving fact-checking accuracy. Refer to Figure 5 for the ROC and Precision-Recall curves illustrating the classifier performance of the best FactGenius variant with two-stage filtering relationship mining and fine-tuned RoBERTa classifier and to Table 3 and 4 for the classification report and confusion matrix, respectively.

Table 2: Comparing our method with other strategies and methods in terms of reported accuracies in the test set. * indicates results obtained from KG-GPT paper (Kim et al., 2023a).

Input type	Model	Variants	One-hop	Conjunction	Existence	Multi-hop	Negation	Total
Claim Only	baseline	Llama3-Instruct-zero-shot	0.61	0.67	0.59	0.61	0.53	0.61
	Fact-KG	BERT*	0.69	0.63	0.61	0.70	0.63	0.65
	KG-GPT	ChatGPT (12-shot)*	-	-	-	-	-	0.68
	baseline	RoBERTa	0.71	0.72	0.52	0.74	0.54	0.68
With Evidence	Fact-KG	GEAR*	0.83	0.77	0.81	0.68	0.79	0.77
	KG-GPT	KG-GPT (12-shot)*	-	-	-	-	-	0.72
	FactGenius	Llama3-Instruct-zero-shot	0.72	0.75	0.76	0.62	0.52	0.68
		BERT-two-stage	0.75	0.67	0.94	0.66	0.79	0.72
		RoBERTa-single-stage	0.87	0.82	0.94	0.75	0.84	0.83
		RoBERTa-two-stage	0.89	0.85	0.95	0.75	0.87	0.85

Table 3: Classification report for the best model across the test dataset.

	Precision	Recall	F1	Support
Refuted	0.81	0.93	0.86	4643
Supported	0.91	0.77	0.83	4398
Accuracy (average)	0.86	0.85	0.85	9041

Table 4: Confusion matrix for the best model across the test dataset.

		Predicted	
		Refuted	Supported
Actual	Refuted	4315	328
	Supported	1031	3367

5 Discussion

The improved performance of FactGenius, particularly in Conjunction, Existence, and Negation reasoning, can be attributed to its innovative combination of zero-shot prompting with large language models and fuzzy text matching on knowledge graphs.

The two-stage approach, which involves an initial filtering phase followed by a validation phase, significantly enhances accuracy. However, the model shows moderate performance improvement in Multi-hop reasoning, indicating the need for more advanced techniques to handle its complexity.

The two-stage approach of filtering and validating connections proved to be particularly effective. In the first stage, the LLM helps to narrow down potential connections based on the

context provided by the claim. This initial filtering significantly reduces the search space, making the subsequent validation stage more efficient. The second stage further refines these connections through fuzzy matching, ensuring that only the most relevant and accurate connections are retained. The comparative study confirmed the importance of each stage, showing that the second stage particularly enhances performance in conjunction and negation reasoning tasks.

As having an LLM inference server is a crucial component of this framework, we employed vLLM (vLLM Project, 2024) to streamline rapid inference with a single NVIDIA A100 GPU. In our experiment, the LLM inference speed was around 15 queries per second, including retries in case of failure. This rate is feasible, considering that LLM inference is continually optimized with the latest technologies. Embedding LLM in a framework has proven to be a wise choice.

6 Conclusion

In this paper, we introduced FactGenius, a novel method that combines zero-shot prompting of large language models with fuzzy relation mining for superior reasoning on knowledge graphs. This approach addresses several key challenges in traditional fact-checking methods. First, the integration of LLMs allows for the leveraging of extensive pre-trained knowledge, which is crucial for understanding and verifying complex claims through structured data from DBpedia. Second, the use of fuzzy text matching with Levenshtein distance ensures that minor discrepancies in entity names or relationships do not hinder the relationship selection process, thus improving robustness.

Our experiments on the FactKG dataset demonstrated that FactGenius significantly outperforms traditional fact-checking methods and existing baselines, particularly when fine-tuning RoBERTa as a classifier. The two-stage approach of filtering and validating connections proved crucial for achieving high accuracy across different reasoning types. This underscores the potential of FactGenius to improve fact-checking accuracy without requiring complex stages and components.

The findings from this study suggest that integrating LLMs with structured knowledge graphs and fuzzy matching techniques holds great promise for advancing fact-checking capabilities. Future work could explore the application of this approach to other domains and datasets, as well as the potential for incorporating additional sources of structured data to further enhance performance.

By improving the accuracy and efficiency of fact-checking, FactGenius contributes to the broader effort of combating misinformation and ensuring the reliability of information in digital communication.

Acknowledgement

This work has benefited from the Experimental Infrastructure for Exploration of Exascale Computing (eX3) at Simula, which is financially supported by the Research Council of Norway.

References

- Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, et al. 2023. [GraphLLM: Boosting Graph Reasoning Ability of Large Language Model](#). *arXiv*.
- Eun Cheol Choi and Emilio Ferrara. 2024. [FACT-GPT: Fact-Checking Augmentation via Claim Matching with LLMs](#). In *WWW '24: Companion Proceedings of the ACM on Web Conference 2024*, pages 883–886. Association for Computing Machinery, New York, NY, USA.
- Kelley Cotter, Julia R. DeCook, and Shaheen Kanthawala. 2022. [Fact-Checking the Crisis: COVID-19, Infodemics, and the Platformization of Truth](#). *Social Media + Society*, 8(1):20563051211069048.
- Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. [ChatLaw: Open-Source Legal Large Language Model with Integrated External Knowledge Bases](#). *arXiv*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). *ACL Anthology*, pages 4171–4186.
- Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, Hao Yang, Andy Kaminski, et al. 2023. [Integrating action knowledge and LLMs for task planning and situation handling in open worlds](#). *Auton. Robot.*, 47(8):981–997.
- Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. [A Survey on Automated Fact-Checking](#). *Transactions of the Association for Computational Linguistics*, 10:178–206.
- Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Supryadi, et al. 2023. [Evaluating Large Language Models: A Comprehensive Survey](#). *arXiv*.
- Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D’amato, Gerard De Melo, Claudio Gutierrez, et al. 2021. [Knowledge Graphs](#). *ACM Comput. Surv.*, 54(4):1–37.
- Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023a. [KG-GPT: A General Framework for Reasoning on Knowledge Graphs Using Large Language Models](#). *ACL Anthology*, pages 9410–9421.
- Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. 2023b. [FactKG: Fact Verification via Reasoning on Knowledge Graphs](#). *ACL Anthology*, pages 16190–16206.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, et al. 2023. [Efficient Memory Management for Large Language Model Serving with PagedAttention](#). In *SOSP '23: Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626. Association for Computing Machinery, New York, NY, USA.
- Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, et al. 2015. [DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia](#). *Semantic Web*, 6(2):167–195.
- Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.
- Xinze Li, Yixin Cao², Liangming Pan, Yubo Ma, and Aixin Sun. 2023. [Towards Verifiable Generation: A Benchmark for Knowledge-aware Language Model Attribution](#). *arXiv*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, et al. 2019. [RoBERTa: A Robustly Optimized BERT Pretraining Approach](#). *arXiv*.
- Robert L. Logan Iv, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. [Barack’s Wife Hillary: Using Knowledge-Graphs for Fact-Aware Language Modeling](#). *arXiv*.
- Meta. 2024. [Meta Llama 3](#). [Online; <https://llama.meta.com/llama3>].

- Preslav Nakov, David Corney, Maram Hasanain, Firoj Alam, Tamer Elsayed, Alberto Barrón-Cedeño, et al. 2021. [Automated Fact-Checking for Assisting Human Fact-Checkers](#). In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, {IJCAI-21}*, pages 4551–4558. International Joint Conferences on Artificial Intelligence Organization.
- Gonzalo Navarro. 2001. [A guided tour to approximate string matching](#). *ACM Comput. Surv.*, 33(1):31–88.
- Stefanos-Iordanis Papadopoulos, Christos Koutlis, Symeon Papadopoulos, and Panagiotis C. Petrantonakis. 2024. [VERITE: a Robust benchmark for multimodal misinformation detection accounting for unimodal bias](#). *Int. J. Multimed. Info. Retr.*, 13(1):1–15.
- Estela Saquete, David Tomás, Paloma Moreda, Patricio Martínez-Barco, and Manuel Palomar. 2020. [Fighting post-truth using natural language processing: A review and open challenges](#). *Expert Syst. Appl.*, 141:112943.
- James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. [FEVER: a Large-scale Dataset for Fact Extraction and VERification](#). *ACL Anthology*, pages 809–819.
- vLLM Project. 2024. [vLLM](#). [Online; <https://github.com/vllm-project/vllm>].
- Yu Wang, Nedim Lipka, Ryan A. Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024. [Knowledge Graph Prompting for Multi-Document Question Answering](#). *AAAI*, 38(17):19206–19214.
- Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, et al. 2024. [Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond](#). *ACM Trans. Knowl. Discovery Data*, 18(6):1–32.
- Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. [KG-BERT: BERT for Knowledge Graph Completion](#). *arXiv*.
- Xia Zeng, Amani S. Abumansour, and Arkaitz Zubiaga. 2021. [Automated fact-checking: A survey](#). *Lang. Linguist. Compass*, 15(10):e12438.
- Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, et al. 2019. [GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification](#). *ACL Anthology*, pages 892–901.

Fact or Fiction? Exploring Diverse Approaches to Fact Verification with Language Models

Tobias Opsahl
University of Oslo
tobiasao@uio.no

Abstract

Recent advancements in natural language processing (NLP) have greatly improved the performance of language reasoning and generating. However, a well known shortcoming of language models is that they tend to generate information that is untrue, referred to as *hallucinations*. In order to improve the correctness of language models, we improve the performance and the computational efficiency of models trained on classifying claims as true or false. We use the FACTKG dataset, which is constructed from the *DBpedia* knowledge graph extracted from Wikipedia. We create fine-tuned text models and hybrid graph and text models that significantly outperform the benchmark FACTKG models, both with respect to test-set accuracy and training time. The increase in performance and efficiency stems from simplifying the methods for retrieving subgraphs, using simple logical retrievals rather than fine-tuned language models. Finally, we construct prompts to ChatGPT 4o that achieves decent performance, but without the need of fine-tuning.

1 Introduction

The field of NLP has greatly improved with the transformer architecture (Vaswani et al., 2017) and vastly scaling up model parameters and training data (Achiam et al., 2023; Bubeck et al., 2023). Large language models (LLMs) trained on a substantial part of all internet data have passed benchmarks as passing the BAR exam (Katz et al., 2024), follow precise and complex coding instructions (Bubeck et al., 2023) and perform data analyst tasks with the same performance as human experts (Cheng et al., 2023). Despite this improvement, state of the art language models still struggle with basic planning (Bubeck et al., 2023) and frequently generates false information, known as *hallucination* (Xu et al., 2024; Huang et al., 2023; Zhang et al., 2023). In order to mitigate hallucination, we

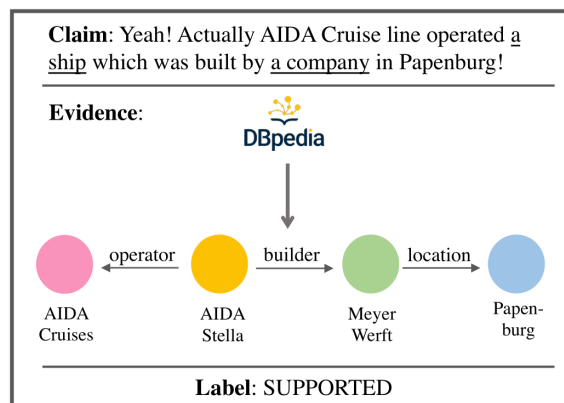


Figure 1: An example claim from FACTKG (Kim et al., 2023). The claim can be verified or refuted based on the DBpedia KG (Lehmann et al., 2015). This is Figure 1 from Kim et al. (2023).

believe it is crucial to be able to classify which information is correct and which is not. Therefore, we dedicate this article to explore models used for fact verification.

One way of structurally working with knowledge is with *knowledge graphs* (KGs). They consist of nodes and edges linked together to represent structural concepts. The *DBpedia* KG (Lehmann et al., 2015) is a large KG extracted from Wikipedia. Nodes represent entities, such as persons, things or events, and edges represent relations, conveying how entities are related, as shown in Figure 1. For instance, a node can be the company *Meyer Werft*, and since it is located in the city *Papenburg*, they are connected with the edge *location*. We refer to *Meyer Werft, location, Papenburg* as a *knowledge triple*.

We propose models trained on FACTKG (Kim et al., 2023), a dataset proposed to better utilise knowledge graphs with fact verification (see Figure 1). It consists of 108 000 English claims that are extracted from the DBpedia knowledge graph. About a third of the claims are manually written,

while the rest are generated from the written claims to be a colloquial form by a language model. The train and validation datasets are equipped with relevant subgraphs for each claim, which one can use to train subgraph retrieval.

In order to work with fact verification, we will work with three main model architectures:

- **Textual Fine-tuning:** Fine-tuning pretrained encoder models on text data for claim verification. We use BERT (Devlin et al., 2018) by concatenating the claims with subgraphs represented as strings.
- **Hybrid Graph-Language Model:** Using a modification of a *Question answer graph neural network* (QA-GNN) (Yasunaga et al., 2021), which both uses a pretrained encoder model to embed the claim, and a GNN to structurally process the subgraphs.
- **LLM Prompting:** Deploying state-of-the-art language models in a few-shot setting, without the need for additional finetuning. We use ChatGPT 4o (Achiam et al., 2023; Open AI, 2024) for this setting.

We selected these three approaches to explore a variety of different models used in NLP, and compare how they perform on fact verification. The text-based finetuning, which is a widely used technique, serves as a conventional method. The QA-GNN architecture is a more specific model for this task, that can efficiently process graph data. We explore various ways to retrieve relevant subgraphs that do not require training of language models, to make the QA-GNN train even more efficiently. In contrast, the LLM prompting displays how general purpose language models can be used for fact verification, without the need of further training.

By utilising efficient subgraph retrieval methods, we are able to substantially increase the test-set accuracy on FACTKG from 77.65% (Kim et al., 2023) to 93.49%. Additionally, our models train quicker, taking only 1.5-10 hours, compared to the 2-3 days spent on the benchmark model from (Kim et al., 2023).

The code and documentation can be found at https://github.com/tobiasao/IN5550_tobiasao/tree/main/home-exam.

2 Related Work

2.1 The FactKG Dataset

The FACTKG dataset (Kim et al., 2023) consists of 108 000 English claims for fact verification, where the downstream task is to predict whether the claim is true or false. The claims are constructed from the DBpedia KG (Lehmann et al., 2015), which is extracted from Wikipedia and represents how entities are related to each other.

The claims are constructed on the following five reasoning types:

- **One-hop:** To answer a one-hop claim, one only needs to traverse one edge in the KG. In other words, only one knowledge triple is needed to verify the validity of the claim.
- **Multi-hop:** As opposed to one-hop claims, one need to traverse multiple steps in the KG to verify multi-hop claims.
- **Conjunction:** The claim includes a combination of multiple claims, which are often added together with the word *and*.
- **Existence:** These claims state that an entity has a relation, but does not specify which entity it relates to.
- **Negation:** The claim contains negations, such as *not*. The generation process varies depending on the reasoning type of the claim.

The dataset is split in a train-validation-test set of proportion 8:1:1. The train and validation set includes relevant subgraphs for each claim, but not the test set. All claims include a list of entities present in the claim and the KG.

2.2 Question Answer Graph Neural Networks (QA-GNNs)

The QA-GNN (Yasunaga et al., 2021) is a hybrid language and GNN model that both uses a pretrained language model to process the text, and couples it with a GNN reasoning over a subgraph. It is given text and a subgraph as input. The text, consisting of a question and possible answers, is added as a node to the subgraph. The language model embeds the text, and assigns a relevance score to each node in the subgraph. The relevance scores are multiplied with the node features, before being sent into the GNN. The GNN output, text-node and the text embedding are concatenated before being put into the classification layer.

3 Methods

3.1 Efficient Subgraph Retrieval

We experiment with different ways of retrieving a relevant subgraph for each claim, focusing on computational efficiency. Each datapoint in the FACTKG dataset consists of a claim and a list of entities that appear both in the claim and the KG. Since the part of DBpedia used in FactKG is fairly large (1.53GB), it is necessary to only use a small subgraph of it as input to the models. The benchmark model from Kim et al. (2023) uses two language models to predict the relevant edges and the depth of the graph. We wish to simplify this step in order to reduce the model complexity, and propose non-trainable methods for subgraph retrieval.

We experiment with the following methods (examples can be found in Figure 2):

- **Direct:** Only includes knowledge triples where both nodes are present in the entity list.
- **Contextualized:** First, include all direct subgraphs. Additionally, lemmatize the words in the claim and check if the nodes in the entity list have any relations corresponding to the lemmatized words in the claim. Include all knowledge triples where at least one node is in the entity list, and the relation is found in the claim.
- **Single-step:** Includes every knowledge triple one can be traversed in one step from a node in the entity list. In other words, include every knowledge triple that contains at least one node in the entity list.

3.2 Finetuning BERT

We use BERT (Devlin et al., 2018) as our pre-trained language model. We fine-tune it with both the claims only, and with claims and various subgraphs. We convert the subgraphs to strings, where each knowledge triple is represented with square brackets, and the name of the nodes and edges are the same as they appear in DBpedia. The order of the knowledge triples are determined by the order of the list of entities in the FactKG dataset and the order of the edges in DBpedia. The subgraphs are concatenated after the claims and a “|” separation token.

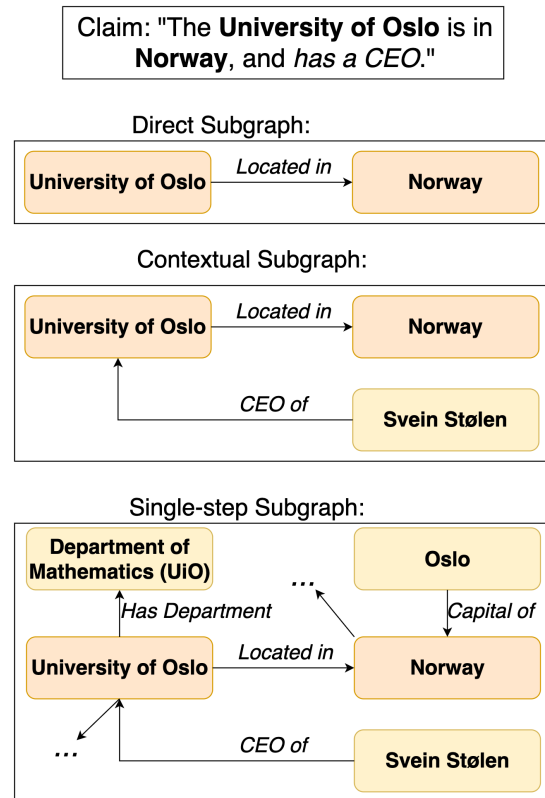


Figure 2: **Examples of the different subgraphs explored in this article.** Boxes and bold letters represent entities, while arrows and *italic letters* represent relations. This claim is meant for illustrative purposes and is not present in the FACTKG dataset.

3.3 QA-GNN Architecture

In order to adapt the QA-GNN to the fact verification setting, we perform some slight modifications. Because the possible answers are always “true” or “false”, we embed only the claims, instead of the question and answer combination. Additionally, we do not connect the embedded question or claim to the subgraph, but use the subgraph as it is from the retrieval.

We use a pre-trained BERT (Devlin et al., 2018) model as the language model to embed and calculate the relevance scores. In order to reduce the complexity of the model, we use a frozen BERT to calculate the embeddings for the nodes and the edges in the graph graph. This way, all of the embeddings in the graph can be pre-calculated. We use the last hidden layer representation of the CLS token, which is of length 768. The BERT that calculates the relevance scores and the embedding of the claim is not frozen. The relevance scores are computed as the cosine similarity between the claim embedding and the embedding of the text in the nodes.

Input Type	Model	One-hop	Conjunction	Existence	Multi-hop	Negation	Total
<i>Claim Only</i>	FACTKG BERT Benchmark	69.64	63.31	61.84	70.06	63.62	65.20
	BERT (no subgraphs)	67.71	67.48	62.51	73.28	64.23	68.99
<i>With Subgraphs</i>	FACTKG GEAR Benchmark	83.23	77.68	81.61	68.84	79.41	77.65
	QA-GNN (single-step)	79.08	74.43	83.37	74.72	79.60	78.08
	BERT (single-step)	97.40	97.51	97.31	80.32	92.54	93.49

Table 1: **Test-set accuracy for the best models from this article and the best benchmark models from (Kim et al., 2023).** The fine-tuned BERT model performed the best, while the QA-GNN was the computationally most efficient model.

We use a graph attention network (Veličković et al., 2017) for our GNN. Since the subgraphs are quite shallow, we only use two layers in the GNN. Each layer has 256 features, which is mean-pooled and concatenated with the BERT embedding and sent into the classification layer. We add dropout (Srivastava et al., 2014) to the GNN layers and the classification layer.

3.4 ChatGPT Prompting

We construct a prompt for ChatGPT 4o in order to answer a list of claims as accurately as possible. This is done by creating an initial prompt and validating the results on 100 randomly drawn claims from the validation set, and by trying different configurations of the prompt until we do not get a better validation set accuracy. We then use the best prompt with 100 randomly drawn unseen test-set questions, and attempt to ask 25, 50 and 100 claims at a time, to see if the amount of claims at a time influences the performance. All the experiments are run three times.

Since we do not have access to vast enough computational resources to run an LLM, this analysis is limited by only using 100 datapoints from the test set. In order to get access to a state-of-the-art LLM, we used the ChatGPT website with a “ChatGPT Plus” subscription to perform the prompting. This model is not seeded, so the exact answers are not reproducible, but every prompt and answer are available in the GitHub repository. We used the ChatGPT 4o model 30th of May 2024. Every prompt was performed in the “temporary chat” setting, so the model did not have access to the history of previous experiments.

Due to the inability to use the entire test set and the lack of reproducibility, we do not directly compare this experiment to the other models. However, we still believe it serves as a valuable benchmark. Recently, the performance of LLMs has rapidly improved, which suggests that their applications will continue to broaden. Additionally, this approach

is not fine-tuned, and may work as an interesting benchmark that can contextualise the results of the other models.

3.5 Benchmark Models

We will compare the results against the best benchmark models from (Kim et al., 2023). The first model is a fine-tuned BERT using only the claims, without incorporating the KG. The model utilising the KG is inspired by GEAR (Zhou et al., 2019), which has been adapted to handle graph-based evidence. It uses two fine-tuned language models to retrieve the subgraphs. One of them predicts relevant edges, the other predicts the depth of the subgraph.

3.6 Further Details

We use an AdamW optimizer (Loshchilov and Hutter, 2017) with a linear learning rate scheduler. We used the UiO USIT ML nodes (University Centre for Information Technology, 2023), where we trained on RTX 2080Ti GPUs.

4 Results: Improved Performance and Efficiency

The test results for our best model configurations and the benchmark models can be found in Table 1. We see that all of our models outperform the benchmark model with respect to test set accuracy. The best performing model is the fine-tuned BERT, followed by the QA-GNN and the benchmark models. The fine-tuned BERT without any subgraphs were able to achieve higher performance than the one from (Kim et al., 2023).

Additionally, our models were much faster to train. While the GEAR model used 2-3 days to train on an RTX3090 GPU (reported by the authors by email), our QA-GNN only used 1.5 hours. The training time of our fine-tuned BERT model was greatly influenced by the size of the subgraphs we used. With no subgraphs, it took about 2 hours to

Model	One-hop	Conjunction	Existence	Multi-hop	Negation	Total
BERT (no subgraphs)	67.71	67.48	62.51	73.28	64.23	68.99
BERT (direct)	80.24	83.30	59.05	77.62	74.58	79.64
BERT (contextual)	81.20	84.45	61.05	77.04	77.40	80.25
BERT (single-step)	97.40	97.51	97.31	80.32	92.54	93.49
QA-GNN (direct)	74.60	74.01	58.97	76.41	74.12	75.01
QA-GNN (contextual)	76.58	69.94	84.68	74.58	80.75	76.12
QA-GNN (single-step)	79.08	74.43	83.37	74.72	79.60	78.08

Table 2: **Test-set accuracy for different subgraph retrieval methods on FACTKG.** The *direct* approach only includes knowledge triples where both nodes appear in the claim, the *contextual* also includes edges appearing in the claim, and the *single-step* includes all knowledge triples where at least one node appear in the claim. The QA-GNN models use the single-step subgraph if the direct or contextual is empty, while the BERT does not.

train, while with the one-hop subgraph it took 10 hours.

4.1 Successful Subgraphs Retrievals

We now look at the different configurations for the subgraph retrievals, which greatly influenced the performance of the models. Since the *direct* and *contextual* approach only includes subgraphs if a certain requirement is fulfilled, they will result in some of the claims having empty subgraphs. In the training and validation set, 49.0% of the graphs were non-empty for the *direct* approach, and 62.5% were non-empty for the *contextual* approach. The *single-step* method resulted in vastly bigger subgraphs.

While the QA-GNN could handle the big subgraphs efficiently, the fine-tuned BERT was severely slowed down when the size of the subgraphs got bigger. Therefore, we substituted any empty subgraphs with the *single-step* subgraph when using QA-GNN, but kept the empty graphs when using fine-tuned BERT.

The results can be found in Table 2. We see a clear improvement in BERT when using the direct subgraphs over none, a small improvement when using the contextual subgraphs, and a big improvement when using the single-step method. The same is true for the QA-GNN, but the differences in performance are smaller.

Since we used non-trainable subgraph retrieval methods and a frozen BERT for embedding the nodes and edges in the subgraphs, we performed this processing before training the models. During training, the models used a lookup table to get the subgraphs and the word embeddings, which significantly decreased the training time. The retrieval of all the subgraphs took about 15 minutes, and the embedding of all the words appearing in them took

about 1 hour. We also tried training a QA-GNN without frozen embeddings, but it ran so slow that we were not able to carry out the training with our available computational resources.

4.2 ChatGPT Works Better when Asking for Explanations

The results for the ChatGPT prompting can be found in Table 3. The accuracy is substantially lower than from our best models, but not too far from the benchmark models. The accuracy is fairly consistent over the three runs, and we do not see a big difference between the amount of questions asked at a time.

Model	Accuracy (mean \pm std)
ChatGPT 25 questions	73.67 \pm 0.5
ChatGPT 50 questions	76.33 \pm 3.3
ChatGPT 100 questions	73.00 \pm 1.4

Table 3: **Test-set accuracy for different configurations of ChatGPT prompting.** The metrics are averaged over three runs. The prompts included 25, 50 or 100 claims at a time, but the same claims were used in all of the configurations.

We started with an initial prompt asking for just the truth values for a list of claims, and updated it to also include some training examples and to ask for explanations. Several configurations of the prompt were tested, and it was also improved based on feedback from ChatGPT.

We saw the biggest improvement when we asked for a short explanation of the answers, instead of just the truth values. Without asking for explanations, the amount of answers were often longer or shorter than the amount of questions, but this never happened when explanations were included. We added numbers to the questions to further help with this issue. We also saw a slight improvement by

Task:
Determine the truth value (True or False) of the following claims based on information verifiable from Wikipedia, as represented in the DBpedia knowledge graph. Provide your answers without using real-time internet searches or code analysis, relying solely on your pre-trained knowledge.

Instructions:

- You will evaluate the following claims, presented one per line.
- Base your answers solely on your knowledge as of your last training cut-off.
- Provide answers in Python list syntax for easy copying.
- Respond with True for verifiable claims, and False otherwise.
- Include a brief explanation for each answer, explaining your reasoning based on your pre-training.
- If the claim is vague or lacks specific information, please make an educated guess on whether it is likely to be True or False.

Output Format: Format your responses as a list in Python. Each entry should be a tuple, formatted as (claim number, answer, explanation).

Example Claims:

1. The Atatürk Monument is located in Izmir, Turkey, where the capital is Ankara.
2. Yes, Eamonn Butler’s alma mater is the University of Texas System!
3. I have heard 300 North LaSalle was completed in 2009.
4. The band Clinton Gregory created an album in the rock style. ...

Example output:

```
[
  (1, True, "The Atatürk Monument is indeed located in Izmir, and the capital of Turkey is Ankara."),
  (2, False, "Eamonn Butler did not attend the University of Texas System; he is a British author and economist whose educational background does not include this institution."),
  (3, True, "300 North LaSalle in Chicago was indeed completed in 2009."),
  (4, False, "Clinton Gregory is primarily known as a country music artist, not rock."),
  ...
]
```

Here are the actual claims you should answer:

Figure 3: **Final prompt used to get truth values from ChatGPT 4o.** The actual questions are not included, but were in the format of the *Example Claims*. The *Example Claims* are from the training set, and the *Example Output* is copy pasted from an actual ChatGPT answer.

formulating the prompt with bullet point lists instead of just paragraphs, and by including some example inputs and outputs from the training set. The final prompt can be found in Figure 3.

5 Discussion

We were able to train better and more efficient models by simplifying the subgraph retrieval methods, both by using a fine-tuned BERT and a slightly modified QA-GNN model. While the QA-GNN models trained the fastest, the fine-tuned BERT clearly performed the best, significantly outperforming the benchmark models.

All of the models perform better the bigger the subgraphs were. This suggests that the model architectures are able to utilise the relevant parts of the subgraphs, without needing an advanced subgraph retrieval step. This is emphasised by our fine-tuned BERT model achieving a 93.49% test set accuracy by only using the single-step subgraphs, while

the GEAR model from (Kim et al., 2023), which trained two language models to perform graph retrieval, achieved a 77.65% test-set accuracy.

One possible limitation of our subgraph retrieval methods is that they never include more than one step away from an entity node, while the trained approach from Kim et al. (2023) is dynamic and may include more. This might make the hypothesis that the simple subgraph retrieval methods will perform worse on *multi-hop* claims than the dynamically trained, however, we see the exact opposite behaviour. The best BERT and QA-GNN models score 80.32% and 74.72% at the multi-hop claims respectively, while the dynamic benchmark model scores 68.84%, even lower than the models not using the subgraphs at all. We do however see that the best performing BERT model clearly performs the worst on the multi-hop claims compared to the other claim types, indicating that even bigger subgraphs might be beneficial.

While the sample size for the ChatGPT metrics were small, it does indicate that non-fine-tuned LLMs can achieve adequate few-shot performance. The performance does not seem to be substantially compromised when the amount of questions prompted increases, so with a bigger access to computational resources, it might be possible to prompt the full test-set at once. The removal of fine-tuning greatly increases the ease of use if one only needs to verify a few claims. Therefore, despite not performing as well as the trained model, this approach could be useful if the performance of LLMs continues to increase.

6 Conclusion and Future Work

Our results show that with simple, yet efficient methods for subgraph retrieval, our models were able to improve with respect to both performance and efficiency. The fine-tuned BERT model with single-step subgraphs clearly achieves the best performance, while the QA-GNN models are more efficient to train.

This indicates that complex models with simple subgraph retrieval methods perform better than the opposite. Since the single-step subgraphs mostly contain information not relevant for the claims, the model is itself able to filter away irrelevant information, and complex subgraph retrieval methods may hence not be necessary for accurate fact verification. Additionally, since the best performing model performed the poorest with *multi-hop* claims, future research could explore simple subgraphs retrieval methods allowing for bigger depths than one.

We also encourage researchers that have access to bigger computational resources to further explore the performance of LLMs for fact verification. A core limitation of our ChatGPT prompting was the inability to use the full test-set, and we consider this crucial for further development. We also think it would be especially interesting to make LLM and KG hybrid models. Since our results indicate that simple single-step subgraph retrievals outperform more complex methods, a promising path would be to simply use both the claims and the single-step subgraphs as input to the LLM. If possible, the LLM could also be fine-tuned on the dataset.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman,

Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrike, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Liying Cheng, Xingxuan Li, and Lidong Bing. 2023. Is gpt-4 a good data analyst? *arXiv preprint arXiv:2305.15038*.

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*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. 2024. Gpt-4 passes the bar exam. *Philosophical Transactions of the Royal Society A*, 382(2270):20230254.

Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. 2023. Factkg: Fact verification via reasoning on knowledge graphs. *arXiv preprint arXiv:2305.06590*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Open AI. 2024. Hello gpt 4o. <https://openai.com/index/hello-gpt-4o/>, Accessed 30.05.2024.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

University Of Oslo University Centre for Information Technology. 2023. Machine learning infrastructure (ml nodes).

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.

- Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.
- Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.
- Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.
- Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemaoy Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren’s song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.
- Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. Gear: Graph-based evidence aggregating and reasoning for fact verification. *arXiv preprint arXiv:1908.01843*.

4 Artificial Text Detection track

Cutting Down to Size: Efficient AI for Artificial Text Detection

Fernando Vallecillos Ruiz
Simula Research Laboratory
Oslo, Norway
fernando@simula.no

Abstract

Nowadays, it is a challenge to distinguish between human or machine-generated written content. This poses a risk to the content integrity and increases the potential for misinformation and misuse. The following study explores the use of structured pruning when classifying texts as human-written or machine-generated on five encoder-only models: BERT, DeBERTa, RoBERTa, XLM-RoBERTa, and DistilBERT. We make use of the SemEval-2024 Task 8 dataset as a framework to fine-tune these models and assess their performance across the different layers. Our findings reveal that pruning can substantially reduce model size without significantly impacting, and even improving, classification accuracy. We demonstrate the potential to create efficient, high-performing models and thus promote sustainable AI practices. The code and materials are available in [the following repository](#).

1 Introduction

The increase of machine-generated text has raised concerns regarding its detection in recent years. Large language models produce human-like text that can become indistinguishable from one written by humans (Zhao et al., 2023). Even if this technology offers benefits, such as content creation or language translation, it can also introduce multiple challenges (Crothers et al., 2023). One of the main challenges is the potential for misuse, for example, the generation of false information, spam, or other forms of content that can deceive people and undermine trust (Uchendu et al., 2023).

Therefore, the capability to detect whether text is human-written or machine-written has become crucial. It has become essential not only to mitigate risks associated with misinformation and misuse, but also for content verification and to maintain quality standards. Research in this area is currently focusing on developing techniques and models able

to perform this binary classification task (Prova, 2024; Abdali et al., 2024). However, there is an ongoing debate about the necessity of using large, resource-intensive models for such tasks. These, often large models, come with a large computational cost and environmental impact to train and deploy (Patterson et al., 2021; Desislavov et al., 2023). Consequently, there is a need to explore more efficient alternatives.

In response to this challenge, this study explores the effectiveness of pruned encoder-only models for the task of binary classification of text as human-written or machine-generated. We employ a structured pruning approach which progressively removes layers from the encoder to evaluate how such reductions affect performance. The study aims to determine if smaller, pruned models can achieve comparable results to their larger counterparts, offering a more sustainable and environmentally friendly solution. Our findings indicate that pruning the models does not necessarily lead to a dramatic decrease in classification accuracy, suggesting that smaller models can achieve comparable results to their larger counterparts.

To this end, Task 8 of SemEval-2024 (Wang et al., 2024) provides an ideal framework. We focus on the monolingual track of Subtask A, which consists of the binary classification of human-written versus machine-generated text on English language. Leveraging the datasets provided, we conduct a thorough evaluation of the five chosen encoder-only models before and after pruning. Furthermore, we stratified the evaluation by text length to determine whether a larger number of neurons is required to accurately classify longer texts.

The primary contributions of this paper are as follows:

- Demonstrating that machine-generated text detection can be achieved with pruned models.
- Analyzing the effects of the input lengths

when detecting whether a text is artificially generated.

This study emphasizes the importance of optimizing the model size for practical applications without compromising performance. Thereby, contributing to the broader discussion on sustainable artificial intelligence practices.

The remainder of this paper is structured as follows. Section 2 explains the problem presented in the SemEval-2024 framework. Section 3 details the methodology, including the chosen models and pruning techniques. Section 4 describes the dataset used. Section 5 details the experimental design and implementation details. Section 6 presents the experimental results and analysis, and discusses the implications of our findings. Section 7 briefly summarizes the related work and Section 8 concludes with a summary of the key contributions.

2 Problem Description

The framework used is the SemEval-2024 Task 8 which focuses on the detection of machine-generated text. This main task is divided into sub-tasks, with Subtask A specifically addressing binary classification of text as either generated by a human or a machine. The primary objective of this subtask is to develop a technique or a models that can classify text into one of two categories: human-written or machine-generated. This binary classification task is crucial for applications such as content verification, misinformation detection, and maintaining the integrity of communications.

Tracks: Subtask A is divided into two tracks: monolingual and multilingual. The monolingual track involves only English sources, while the multilingual track extends to 8 different languages. This study focuses on the monolingual track.

Dataset: This subtask provides a comprehensive dataset for training and evaluation. The training data consists of 119,757 examples that cover a balanced mix of human-written and machine-generated texts. The development set contains 5,000 examples, equally balanced, to evaluate the performance of the models. In a similar manner, the test set contains 36,000 examples.

Evaluation Metric: The primary metric for evaluating the models in Subtask A is accuracy. Accuracy measures the proportion of correct predictions out of the total number. This metric is straightforward and provides a clear-cut indication on the effectiveness of the technique.

3 Methodology

We have chosen to address the binary classification task of distinguishing between human-written and machine-generated text using encoder-only models. The selected models are the following: BERT ¹ (Devlin et al., 2019), DeBERTa ² (He et al., 2020), RoBERTa ³ (Liu et al., 2019), XLM-RoBERTa ⁴ (Conneau et al., 2020), and DistilBERT ⁵ (Sanh et al., 2020). They are known for their strong performance in various NLP tasks.

To fine-tune the models, we have used the training dataset described in the previous section, which consists of 120,000 examples labeled as human-written or machine-generated. The models have been fine-tuned using their default hyperparameters to ensure consistency and comparability.

We implemented structured pruning to explore the impact of model size on performance. The processes consist of incrementally removing layers from the encoder, starting from the last layer, and then assessing the effects on the chosen metric, accuracy. This process aims to determine if smaller models can maintain comparable performance to their non-pruned counterparts. Furthermore, we have stratified the evaluation results by text length. This division aims to determine if longer texts require higher number of neurons, and therefore more complex models, to maintain classification accuracy. Further experimental details are provided in Section 5.

The goals of this work is to contribute to the development of efficient and sustainable methods for artificial text detection. To this end, the objectives of the study are as follows.

- To assess the impact of structured pruning on model performance and determine if models size can be reduced without significant loss of accuracy.
- To analyze the effect of text length on classification performance and determine if longer texts require more complex models.

4 Dataset and Preprocessing

The training dataset contains 119,757 samples of text of which 63,351 are generated by humans and

¹BERT Model (Hugging Face)

²DeBERTa Model (Hugging Face)

³RoBERTa Model (Hugging Face)

⁴XLM-RoBERTa Model (Hugging Face)

⁵DistilBERT Model (Hugging Face)

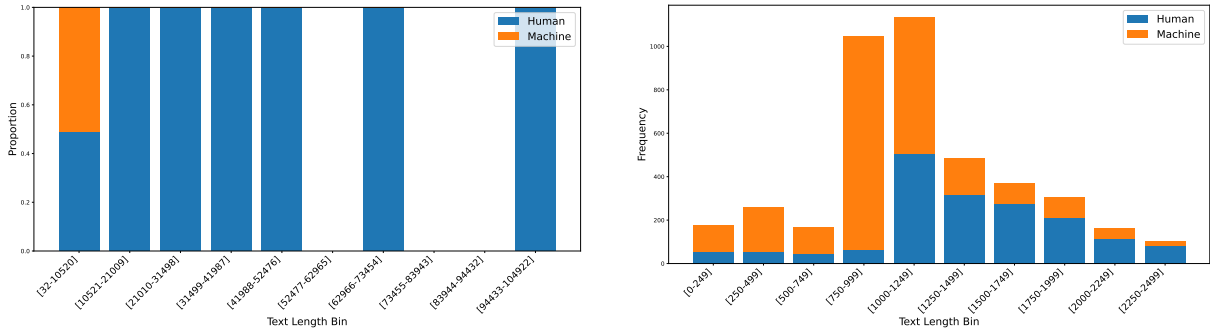


Figure 1: Distribution of labels stratified by text length.

56,406 are generated by a machine. Although there is a test set, as described in the previous section, we have decided not to use it in the following experiments. Instead, we discarded the original test set and treated the development set as a test set. This development set, referred to as test set from now on, contains 5,000 samples of text, half generated by humans and half by a machine. The reason being that the goal of this study is to show the possibility of pruning models to achieve similar results, not to propose one single model that can overcome the rest.

Our new test set, i.e. the original development set, is used to evaluate the accuracy of the model and to further analyze the results. The distribution of labels when stratifying the test set can be seen on the left of Figure 1. As it can be appreciated, the distribution of labels is extremely skewed, since all machine-generated examples are shorter. This is likely due to the context window by which most models are restricted.

To perform a more realistic analysis, we create a subset by imposing a limit on the text length of 2,500. This arbitrary division results in a more balanced set that contains 2,491 examples of machine-generated text and 1,722 examples of human-generated text. The distribution can be seen on the right of Figure 1.

5 Experimental Design

5.1 Model and Training Configurations

The following encoder-only models are selected for this study: BERT, DeBERTa, RoBERTa, XLM-RoBERTa, and DistilBERT. Each model and tokenizer were configured with its default settings as provided by the respective implementations in the transformers library. The selection of these models was driven with the intent of covering a wide range of common encoder-only models, with

DistilBERT included as a smaller condensed model for comparative purposes.

5.2 Pruning Strategy

The layers were pruned incrementally, one by one, starting from the last layer of the encoder. The models were evaluated at each increment to assess the performance drop, if any, and to determine the minimum amount of layers required to maintain accuracy.

5.3 Stratification by Text Length

Longer texts may require a higher number of neurons and more complex models to maintain accuracy. To explore the effect of text length on model performance, we stratified the evaluation results by text length into 10 bins of the same size. The inputs were categorized into different length ranges, and accuracy scores were calculated for each range.

5.4 Implementation Details

The experiments were implemented using Python 3.10, PyTorch, and the transformers library. The models were fine-tuned using the following parameters:

- **Batch size:** 8
- **Number of epochs:** 5
- **Learning rate:** 2×10^{-5}
- **Weight decay:** 0.01

The fine-tuning process was implemented using the Trainer class from the transformers library. The models were trained on an NVIDIA A100 GPU.

By following this experimental design, we aim to provide a thorough evaluation of encoder-only models for binary classification of human-written

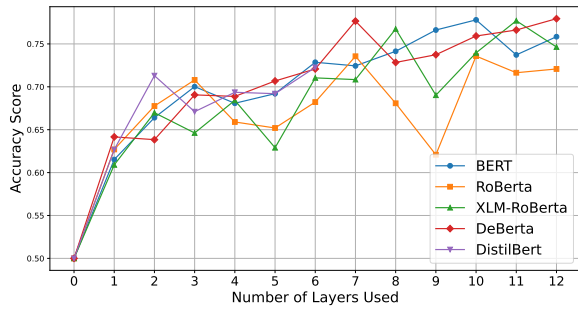


Figure 2: Average accuracy scores for each model.

versus machine-generated text. We also aim to explore the potential to reduce the size of the model through pruning without compromising accuracy.

6 Results and Discussion

6.1 Pruned Models

The pruning process reveals insights and trends about the performance of the models when the number of layers used is decreased. Each model was evaluated from using all their layers to none of the layers in the encoder.

BERT displayed the highest accuracy score of 0.78 with 10 layers, which indicates that slightly reducing the model size can enhance performance. This may suggest that some layers may introduce redundancy or noise. Even with further pruning down to 7 or 8 layers, BERT is able to maintain high accuracy, demonstrating its robustness and efficiency.

On the other hand, RoBERTa showed a slightly different trend. Although it also achieved high accuracy scores with reduced layers, its performance was more variable. The model accuracy peaked at 7 layers with a score of 0.74, but a notable dip occurred at 9 layers, where the accuracy dropped to 0.62 before recovering. The variability shown highlights how identifying specific pruning points that can optimize performance is crucial.

XLM-RoBERTa displays stable performance through the different pruning levels. It achieved its highest accuracy of 0.78 with 11 layers, and maintained strong performance with fewer layers, such as 8 layers where it scored 0.77.

DeBERTa demonstrated high accuracy throughout the pruning process, peaking at 12 layers with an accuracy of 0.78. Even with pruning, DeBERTa maintained competitive accuracy, scoring 0.77 with 11 layers and 0.74 with 9 layers.

DistilBERT, as a model already designed to be

compact, achieved its highest accuracy of 0.72 with 6 layers, i.e. all its original layers. The performance of this model underscores the effectiveness of smaller architectures in maintaining accuracy while offering the benefits of reduced computational costs. Our results show that even smaller distilled models may benefit from a reduction in size. The stable performance of DistilBERT with fewer layers displays the potential for deploying lightweight models.

Overall, these results illustrate that pruning can substantially reduce model size while maintaining or even improving classification accuracy. Each model showed that there are optimal pruning points where performance is similar despite reduced complexity. This can offer computational and environmental benefits. Furthermore, it also improves the practical applicability of these models in real-world scenarios. Identifying these optimal pruning points may lead to more efficient models, which are computationally less expensive and environmentally friendly. Future research can delve into more nuanced pruning strategies and their effects on a broader range of tasks and models.

6.2 Stratification by Text Length

The performance of all pruned models was further evaluated based on the different lengths of the inputs. Figure 3 provides the accuracy scores for each model with varying numbers of layers used and ranges of text length.

BERT’s accuracy increased across all text length ranges as more layers were used. The performance was interesting for longer texts (i.e., 2000-2249 tokens), where the accuracy reached 0.70 with no layers and improved to 0.83 with 10 layers. Shorter texts (i.e., 0-249 tokens) also saw significant improvement, achieving an accuracy of 0.97 with 10 layers.

RoBERTa’s performance was less stable in regards with text length. While shorter texts (0-249 tokens) saw improved accuracy up to 0.74 with 10 layers, the model’s accuracy for slightly longer texts (750-999 tokens) was less stable, varying with different layers. This may be due to the fact that text length plays a significant role in its classification accuracy. Therefore, the model may benefit from pruning strategies that are based on text length.

XLM-RoBERTa showed good and stable performance throughout different text lengths and pruning levels. The model consistently achieved high

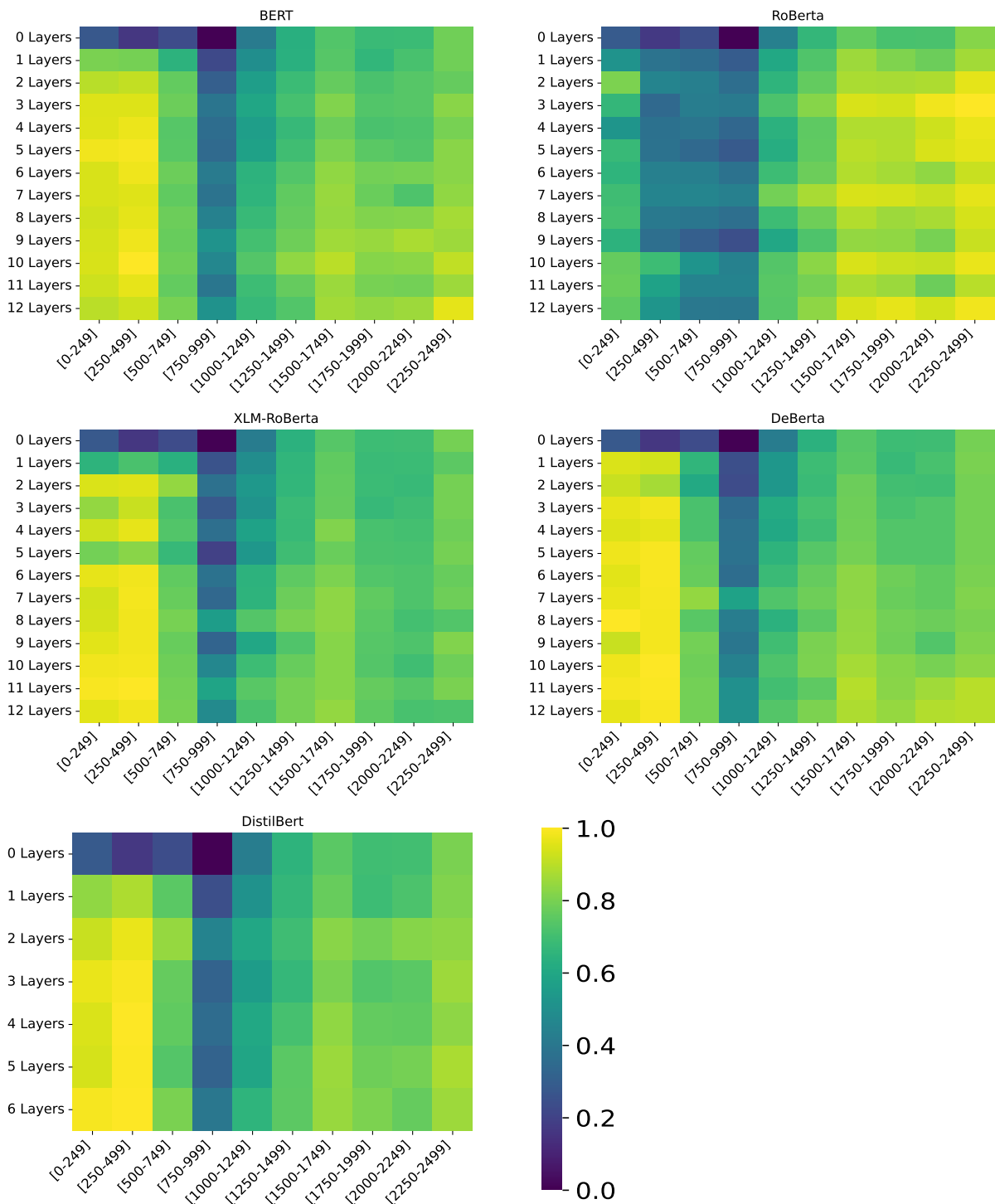


Figure 3: Accuracy scores per model stratified by length.

accuracy for shorter texts (0-249 tokens), peaks at 0.98 with 11 layers, and maintained strong performance for longer texts (2000-2249 tokens), reaching 0.74 with 11 layers.

DeBerta’s scores were high across all text length ranges and pruning levels, too. The model reached its peak accuracy of 0.88 with 11 layers for texts in the range of 1500-1749 tokens. We can conclude that DeBerta is effective in handling both short and long texts with minimal performance degradation.

DistilBERT, although a smaller and distilled model, showed good performance through the different text lengths. The accuracy for shorter texts (0-249 tokens) peaked at 0.96 with 6 layers. For longer texts (2000-2249 tokens), the model achieved a high accuracy of 0.78 with 5 layers.

In summary, the stratification by text length and its analysis underscore the robustness of these models in handling different text sizes. However, there is an undeniable trend of performance improvement with a higher number of layers. This may suggest that, although pruning reduces model complexity, maintaining certain number of layers is crucial to achieve high accuracy through different text lengths. These results reinforce how important it is to balance model size and performance, especially when the classification relates to texts with diverse lengths.

6.3 Comparison of Accuracy Differences with Half Layers Discarded

Lastly, we compared the accuracy differences between the full model and when pruning half of the layers for each model in Figure 4. Positive differences show a decrease in accuracy, while negative differences indicate improved performance with fewer layers. This can be used as a straightforward and baseline strategy of structured pruning.

On average, all models display a decrease in accuracy when half of the layers are discarded, which is to be expected since pruning typically reduces model capacity. For example, BERT saw its accuracy decrease by 0.123 for texts in the range of 2250-2499 tokens, suggesting a substantial impact when layers are pruned. RoBERTa also displayed a notable decrease of 0.091 for shorter texts (0-249 tokens), further proving the general trend of performance reduction with fewer layers. However, some models showed less substantial decreases or even slight improvements in performance for certain text lengths. XLM-RoBERTa demonstrated robustness and stability with a slight increase in accuracy of

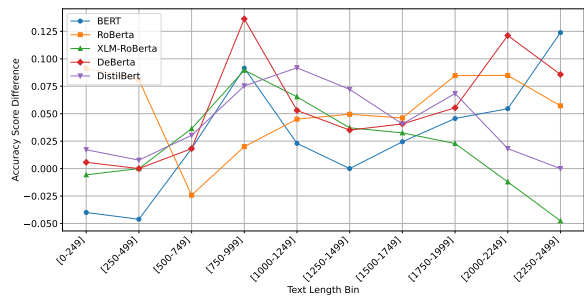


Figure 4: Difference in accuracy scores for each model when half of the layers are removed. Positive differences indicate a decrease in accurate and negative differences show improved performance with half of its layers.

0.0057 for texts in the range of 0-249 tokens.

Overall, these results suggest that although pruning generally reduces accuracy, it can also lead to improvements in specific scenarios. This experiment highlights the potential for optimizing model architectures through more targeted pruning strategies, balancing model complexity and performance.

7 Related Work

7.1 Artificial Text Detection

Artificial text detection has gained significant traction due to the increase of synthetic text generated by advanced models such as GPT-4 and other large-scale language models (Wu et al., 2024). Previous studies have approached the problem from different perspectives to try to detect the differences between text generated by a human or a machine. A prominent and common method involves machine learning classifiers trained on features extracted from text. This can include linguistic patterns, lexical distributions, and syntactic structures (Liu et al., 2015; Hassan et al., 2015). Other approaches have focused on deep learning techniques such as recurrent neural networks and transformer models (Zellers et al., 2020; Ruchansky et al., 2017). These models directly learn the characteristics of artificial text from large datasets. Furthermore, some studies explore the effectiveness of ensemble and mixture techniques, in which multiple classifiers are combined to improve detection accuracy (Ruchansky et al., 2017). However, the challenge of adapting detection methods to increasingly large and more sophisticated text generation models still remains.

7.2 Pruned Models

Pruning is a common technique in machine learning to reduce model size and computational cost without substantially compromising performance (Han et al., 2015). These pruned models can reduce the number of parameters, leading to faster inference times and lower resource consumption (Cheng et al., 2023). Studies have shown that pruned versions of neural networks retain substantial discriminatory power, suggesting a viable route to deploy lightweight models in resource-constrained environments (Frankle and Carbin, 2019).

There are two primary pruning techniques: structured pruning, which removes entire units within a network and changes the model’s shape, and unstructured pruning, which maintains the model’s shape but introduces zeros within the units. Advancements in both techniques have shown promising results enhancing model efficiency while maintaining their performance (Lu et al., 2024).

8 Conclusion

This study evaluated the impact of structured pruning on five encoder-only models to distinguish between text generated by a machine or a human. Our findings show that pruning can significantly reduce model size while maintaining or even improving classification accuracy. Models such as BERT and DeBERTa were benefitted from reduced layers, which may indicate the removal of redundancy or noise. However, the impact of pruning was not consistent, with other models, like RoBERTa, displaying more performance variability.

The stratification by text length revealed insights on the importance of maintaining certain number of layers to keep high accuracy across various text lengths. Furthermore, the comparison of accuracy differences when directly discarding half of the layers demonstrated that even straightforward pruning approaches can offer substantial benefits. These insights underscore the potential for optimizing model architectures to balance complexity and performance, conducting to more efficient and environmentally-friendly models.

This research highlights the feasibility of structured pruning as a viable approach for sustainable AI, reducing environmental and computational costs without substantially compromising effectiveness. Future work could investigate more complex pruning techniques through a wider range of models and tasks to further maintain efficiency.

References

- Sara Abdali, Richard Anarfi, C. J. Barberan, and Jia He. 2024. [Decoding the AI Pen: Techniques and Challenges in Detecting AI-Generated Text](#). *Preprint*, arxiv:2403.05750.
- Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2023. [A Survey on Deep Neural Network Pruning-Taxonomy, Comparison, Analysis, and Recommendations](#). *Preprint*, arxiv:2308.06767.
- 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.
- Evan Crothers, Nathalie Japkowicz, and Herna Viktor. 2023. [Machine Generated Text: A Comprehensive Survey of Threat Models and Detection Methods](#). *Preprint*, arxiv:2210.07321.
- Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. 2023. [Compute and Energy Consumption Trends in Deep Learning Inference](#). *Sustainable Computing: Informatics and Systems*, 38:100857.
- 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.
- Jonathan Frankle and Michael Carbin. 2019. [The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks](#). *Preprint*, arxiv:1803.03635.
- Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. [Learning both Weights and Connections for Efficient Neural Networks](#). *Preprint*, arxiv:1506.02626.
- Naemul Hassan, Chengkai Li, and Mark Tremayne. 2015. [Detecting Check-worthy Factual Claims in Presidential Debates](#). In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM ’15*, pages 1835–1838, New York, NY, USA. Association for Computing Machinery.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. [DeBERTa: Decoding-enhanced BERT with Disentangled Attention](#). In *International Conference on Learning Representations*.
- Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. [Real-time Rumor Debunking on Twitter](#). In *Proceedings of the*

- 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, pages 1867–1870, New York, NY, USA. Association for Computing Machinery.
- 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](#). *Preprint*, arxiv:1907.11692.
- Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. [Not All Experts are Equal: Efficient Expert Pruning and Skipping for Mixture-of-Experts Large Language Models](#). *Preprint*, arxiv:2402.14800.
- David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluís-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. [Carbon Emissions and Large Neural Network Training](#). *Preprint*, arxiv:2104.10350.
- Nuzhat Prova. 2024. [Detecting AI Generated Text Based on NLP and Machine Learning Approaches](#). *Preprint*, arxiv:2404.10032.
- Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. [CSI: A Hybrid Deep Model for Fake News Detection](#). In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. [DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter](#). *Preprint*, arxiv:1910.01108.
- Adaku Uchendu, Thai Le, and Dongwon Lee. 2023. [Attribution and Obfuscation of Neural Text Authorship: A Data Mining Perspective](#). *ACM SIGKDD Explorations Newsletter*, 25(1):1–18.
- Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Chenxi Whitehouse, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024. [SemEval-2024 Task 8: Multidomain, Multimodel and Multilingual Machine-Generated Text Detection](#). *Preprint*, arxiv:2404.14183.
- Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Derek F. Wong, and Lidia S. Chao. 2024. [A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions](#). *Preprint*, arxiv:2310.14724.
- Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2020. [Defending Against Neural Fake News](#). *Preprint*, arxiv:1905.12616.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen

Impact of Domain Variability in Training Data for Monolingual Artificial Text Detection

Dans Reinicans
dansr@ifi.uio.no

Abstract

Artificial text detection is the task of identifying whether the given text is generated by a language model or written by a human. This paper analyzes how the absence of texts written or generated in a particular domain in the training set will influence the model's performance in distinguishing between human-written text and language model-generated text. The results demonstrate which domains and which models used to generate text for training data are the most and least impactful.

1 Introduction

Artificial text detection is a task in the field of Natural Language Processing (NLP) that focuses on identifying whether a given text is generated by a human or an artificial intelligence, such as a language model. With the growing popularity of Large Language Models (LLM), the amount of machine-generated texts also increased. With that, research on artificial text detection focuses on several essential tasks, such as fake news or plagiarism detection. Much research has also been done on the topic. For instance, a competition organized by Yuxia Wang, Jonibek Mansurov, and Preslav Nakov from Mohamed Bin Zayed University of Artificial Intelligence showcases different methods used to fine-tune LLMs or other methods in order to distinguish between AI-generated text and human-generated text. (Wang et al., 2024a) One of the tasks of the competition focused on developing the best model for monolingual artificial text detection, which focuses solely on English texts.

One of the challenges when developing a robust artificial text detection model is its generalizability across different domains. For example, a model trained on a dataset consisting of social media posts or fiction novels might struggle to determine whether a machine or a human writes a scientific text or a news article. This domain-specific performance means the model must be trained across a

variety of domains in order to enhance its adaptability and overall performance.

This paper investigates the importance and influence of different domains in training data for monolingual texts, specifically English texts. It specifically examines the AI-text detection dataset (Wang et al., 2024b), analyzing which domains present better training data for classification tasks and how different text generated by different models also influences the different model's overall performance.

The structure of the paper is as follows: Section 2 describes the data used for experiments. Section 3 provides an overview of the models chosen for testing. Section 4 outlines the experiments conducted. Section 5 summarizes the findings and results. Conclusions are given in the Section 6. Section 7 concludes the paper with the suggestions for future work.

2 Data

This paper makes use of the extension of M4, a large-scale multi-generator, multi-domain, and multi-lingual corpus, for detecting machine-generated texts in a black-box scenario where there is no access to a potential generator or its outputs except for plain text. (Wang et al., 2024b) The data used for this experiment focuses on the English language from five domains: Wikipedia, Reddit ELI5, WikiHow, PeerRead, and arXiv abstract. The domains of these sources range from highly formal and technical, represented by arXiv abstract and PeerRead, to informal, represented by Reddit ELI5, with Wikipedia and Wikihow falling somewhere in between, where Wikipedia represents an encyclopedia-type data, and Wikihow represents an instructional how-to guide. This dataset contains parallel data of texts generated by several text-generating models in the given domains. These models are Davinci003, ChatGPT, Cohere, Dolly-

v2 (Conover et al., 2023), and BLOOMz (Muenighoff et al., 2022). Davinci003, ChatGPT, Cohere, and Dolly-v2 were used to generate texts for the sources in the training dataset, and BLOOMZ was used to generate texts for the sources in the test dataset. The distribution of data per dataset, source, and model can be seen in Table 1, and an example of the instance in the dataset is in Table 2.

3 Models

For the task of artificial text detection, this paper used several language models to evaluate how well these models can distinguish between AI-generated text and human-written text. The first model used is the BERT (Bidirectional Encoder Representations from Transformers) language model introduced by (Devlin et al., 2019). Specifically, I used BERT-base, a transformer-based model pre-trained on large text corpora tasks of Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). It is mainly pre-trained using full texts from Wikipedia articles, covering a variety of topics, linguistic structures, and a broad range of information across different areas of knowledge and domains.

A second model used for this task is RoBERTa (A Robustly Optimized BERT Pretraining Approach), a base case model developed by Facebook AI (Liu et al., 2019). RoBERTa is also a transformer-based model with 12 transformer layers, 768 hidden dimensions, and 12 attention heads. It is built upon the BERT model by optimizing the pretraining strategies, for example, using more diverse datasets containing news articles, books, and Wikipedia articles.

The third model used is the RoBERTa-based OpenAI Detector developed by OpenAI (Solaiman et al., 2019). This is also a transformer-based language model. This model is pretrained explicitly on the outputs of GPT-2 model-generated texts. Its purpose is to predict whether a text is generated by the GPT-2 model.

The last model used is the RoBERTa mixed detector by (Sivesind and Winje, 2023). This is a transformer-based language model. It is pre-trained on 15,000 datapoints of GPT-wiki-intros and 10,000 ChatGPT-Research-Abstracts. The model's purpose is to determine whether the text is generated by a language model similar to ChatGPT and GPT-3 or whether it is a human-written text.

The motivation for choosing these four models is that in the competition mentioned before, the

best-performing models are transformer-based, so all the chosen models are also transformer based. The specific motivation for choosing the last two models is to observe the results of a different model, trained and fine-tuned on different data sets, compared to the first two models, to distinguish between the human written text and a text generated by a Language model. As there is no direct comparison of performance between two similar models trained on the different datasets, the author decided to explore whether a model like RoBERTa-based OpenAI Detector, pre-trained to detect GPT-2 generated texts, could be used for the purpose of distinguishing between the texts generated by humans or any other model and see the model performance or how well a model trained to predict texts generated by GPT type models can be used to predict texts generated by other models. The main goal of this paper is to observe how the performance of the models differs based on the data provided in the training dataset and not achieve a result that can compete in the leaderboards.

4 Experimentation

This paper carries out four experiments, each with a different model. Subsection 4.1 concerns fine-tuning the BERT-base model for detecting machine-generated texts. Subsection 4.2 describes fine-tuning the RoBERTa base-case for the detection of machine-generated texts. Subsection 4.3 describes the prediction generation for the test set using the RoBERTa-based OpenAI Detector. Finally, subsection 4.4 generates the predictions using the RoBERTa mixed detector.

4.1 BERT-base

The goal of this experiment is to fine-tune the BERT-base model using different sets of training data. There are a total of 9 instances of training resulting in 9 differently fine-tuned BERT-base models. The data is first read from the JSON file, omitting either the source (Wikipedia, Wikihow, Peer-read, Reddit, or arXiv) or the model generating the text (davinci-003, ChatGPT, Cohere, Dolly-v2). Afterwards, the data is tokenized and split into 80% train and 20% development datasets. After pre-processing the data, the fine-tuning phase starts. The model is fine-tuned with these hyperparameters: learning rate of $2e-5$, batch size of 16, number of training epochs of 3, and weight decay of 0.01. After the fine-tuning process for three epochs is

Split	Source	davinci-003	ChatGPT	Cohere	Dolly-v2	BLOOMz	Machine	Human
Train	Wikipedia	3,000	2,995	2,336	2,702	-	11,033	14,497
	Wikihow	3,000	3,000	3,000	3,000	-	12,000	15,499
	Reddit	3,000	3,000	3,000	3,000	-	12,000	15,500
	arXiv	2,999	3,000	3,000	3,000	-	11,999	15,498
	PeerRead	2,344	2,344	2,342	2,344	-	9,374	2,357
Dev	Wikipedia	-	-	-	-	500	500	500
	Wikihow	-	-	-	-	500	500	500
	Reddit	-	-	-	-	500	500	500
	arXiv	-	-	-	-	500	500	500
	PeerRead	-	-	-	-	500	500	500

Table 1: **Subtask A: Monolingual Artificial Text Detection.** Data statistics over Train/Test splits. (Wang et al., 2024a)

id	identifier of the example
label	label (human text: 0, machine text: 1)
text	text generated by a machine or written by a human
model	model that generated the data
source	Wikipedia, WikiHow, Reddit, arXiv, and PeerRead

Table 2: Example of an instance in the train dataset

complete, the model is saved for later use in generating the predictions. After the model is saved, the model is loaded in order to show the metrics of the model’s performance and generate the predictions for either the development or test set. After the predictions are generated, they are saved in the form specified before as a JSON file. With the predictions ready and saved, they are used together with the test set to produce scores that can be analyzed. The scores produced are f1-macro, f1-micro, and accuracy.

4.2 RoBERTa base cased

Similar to the previous experiment, this experiment’s goal is to fine-tune the RoBERTa-based model using different sets of data. Again, there are a total of 9 differently fine-tuned RoBERTa-base-case models on the same nine sets of training data. The same sources and models as in the previous experiment are omitted from each training set. The training development split of data stays the same, with 80% of training data used for training and 20% of data used as development set. The same method of fine-tuning the model using a Trainer class with hyperparameters: learning rate of $2e-5$, batch size of 16, number of training epochs of 3, and weight decay of 0.01 is used again. After the model is fine-tuned and saved, it is used to generate predictions and save them for the test set. In the

end, the same metrics are used to get and analyze the result: f1-macro, f1-micro, and target metric accuracy.

4.3 RoBERTa base OpenAI detector

The difference between this experiment and the previous one is that this model did not undergo the same fine-tuning process as the models before that. It is pre-trained on the data mentioned in section 3 and used as it is to generate predictions for the test set. Afterwards, the predictions are evaluated on the previously mentioned metrics: f1-macro, f1-micro, and accuracy.

4.4 RoBERTa-mixed-detector

As in a previous subsection, this model is not fine-tuned on any of the train set data and is used straight as is to generate the predictions for the test data set. The predictions generated and saved are evaluated using the same metrics: f1-macro, f1-micro, and accuracy.

5 Results

This section provides a report and reviews the results of the experiments detailed in the Section 4.

5.1 BERT-base

Results for the BERT-base model can be seen in the Table 3. The table mentions metrics used for evaluation and the source/domain or model texts that are not used in the fine-tuning process. The target metric for the model’s performance is accuracy. The first line in the table is the baseline, where all the sources and domains are present in the training set and are used to train the model. According to the results presented in the table, the least influential resource for artificial text detection in the case of the BERT-base model is PeerRead, as it has produced the highest accuracy of 0.831, meaning its absence from the training data did not lower the accuracy drastically, but even increased it in comparison to the baseline. The most influential source of information is Wikipedia, which produced the lowest score when absent in the training data. When Wikipedia is not used, the model’s performance drops drastically. In terms of models, the model that least affected the performance by being absent from the training data is Cohere, with an accuracy score of 0.788, and the model whose absence has most negatively impacted the performance on the test set, is Dolly-v2, with an accuracy score of 0.718. It is also worth noting that by excluding arXiv, PeerRead, Reddit, and Cohere, the performance improved compared to the baseline.

Interestingly, excluding WikiHow as a domain and chatGPT as a model from the training data produced identical accuracy scores of 0.749. The only difference lies in F1-macro scores for both models. With WikiHow excluded from the training data and achieving an F1-macro score of 0.743, and ChatGPT, also excluded, achieving an F1-macro score of 0.741, it appears that the model fine-tuned without ChatGPT data can better distinguish between human-written and AI-generated texts. Additionally, not using WikiHow articles in the fine-tuning process results in a more balanced and consistent performance across both types of texts.

5.2 RoBERTa base cased

Results for the RoBERTa-base-cased model can be seen in the Table 4. The table is built in the same way as in the previous subsection, with the domain or model excluded from the training data in the first column and F1-macro, F1-micro, and accuracy metrics in the subsequent columns. The baseline results with all the models and sources present in the training data are in the first line of the table. The

target metric is accuracy. According to the results in Table 4, the highest accuracy score is again with the model that did not have a PeerRead human written and AI-generated text in its training data. The accuracy score for this model is 0.858. This model outperforms the baseline by quite a lot: 0.15 points. The domain whose absence in the training data impacted the drop in accuracy the most is arXiv this time, with an accuracy score of 0.615. This is the lowest accuracy score across all domains and models for both BERT-base and RoBERTa-base-cased models fine-tuned on the training data. In terms of models used for text generation, the model whose absence impacted the performance the least is ChatGPT, with a score of 0.719, and the model whose absence impacted the performance of the RoBERTa-base-cased the most is Davinci-003, with an accuracy score of 0.648. In general, the RoBERTa-base-cased performed worse in comparison to the BERT-base. There is the same tendency as in the previous subsection. By excluding either PeerRead texts or Reddit texts from the training data, the model achieves a better result than the baseline. There are more than half of the scores lower than 0.70 accuracy in the RoBERTa-base-cased model, while the BERT-base had accuracy scores above the 0.70 mark. At the same time, the RoBERTa-base-cased model produced the overall best result of 0.858 across all the models experimented within this paper, with PeerRead absent in the training data. Overall, comparing BERT-base and RoBERTa-base-cased models, the BERT model performed better by around 0.05 - 0.1 points in accuracy, with the exception of PeerRead and Reddit, where RoBERTa had a superior accuracy score.

5.3 RoBERTa base OpenAI detector and RoBERTa-mixed-detector

As mentioned before, these two models were not fine-tuned on any of the training data like the models in subsections 5.1 and 5.2. The results for these two models can be seen in the Table 5. RoBERTa base OpenAI detector and RoBERTa-mixed-detector have accuracy scores of 0.493 and 0.345, respectively. The baseline for both BERT and RoBERTa models from previous subtasks outperforms both RoBERTa base OpenAI detector and RoBERTa-mixed-detector, resulting in them being the two worst-performing models to be experimented with in this paper. One of the reasons for such a weak performance in terms of accuracy

score might lay with the training data. RoBERTa-based OpenAI Detector was trained only using the outputs of the GPT2 model, and the RoBERTa mixed detector also used data points generated by GPT-type models.

6 Conclusion

This paper analyzes the effects of using different sets of training data by excluding one of the domains of AI-generated or human-written texts or models used to generate text in the before-mentioned domains from the training data and fine-tuning different transformer-based models using this training data in order to distinguish between an AI-generated text and human-written text. By using the extension of the M4 dataset and BERT-base and RoBERTa-base-based models, it is found that the domain that is the least impactful in differentiating between AI-generated texts and human-written texts is PeerRead. One of the problems is correctly identifying whether it is the highly former and scientific domain that is least useful for distinguishing between human-written texts or AI-generated texts or the fact that PeerRead contains the least amount of text in the dataset. By leaving PeerRead absent from the training dataset, it does not affect the amount of data contained in the dataset, like leaving out, for example Reddit texts, which are around 13,000 and 28,000 texts, respectively. The model least helpful in distinguishing between human texts and AI-generated texts is ChatGPT, as its absence in the training data produced the highest average accuracy scores, and the most useful model is Dolly-v2, as its absence in the training data hindered the accuracy scores the most.

Moreover, based on accuracy scores from two manually fine-tuned models and two pre-trained models, it has been discovered that a much better performance is generally achieved with a wider variety of domains and models used in training data. Though there are some exceptions, as was shown in comparison to the baseline by omitting some of the sources or models, it is possible to achieve a better result than just using all the available data. This finding could possibly be applied to other areas of study within artificial text generation.

7 Future work

Given that this model tested only training datasets that excluded one domain or one model per training iteration, it would be interesting to see how

combinations of different domains, models, or domains and models would influence the accuracy with which fine-tuned models can distinguish between AI-generated texts and human-written texts.

References

- M. Conover, M. Hayes, A. Mathur, J. Xie, J. Wan, S. Shah, A. Ghodsi, P. Wendell, M. Zaharia, and R. Xin. 2023. [Free dolly: Introducing the world's first truly open instruction-tuned llm](#).
- J. Devlin, M. Wei, C. Kenton, and L. Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#). *Proceedings of NAACL-HLT 2019*, pages 4171–4186.
- Y. Liu, 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](#).
- N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. Le Scao, M. Saiful Bari, S. Shen, Z. Xin Yong, H. Schoelkopf, X. Tang, D. Radev, A. Fikri Aji, K. Almubarak, S. Albanie, Z. Alyafeai, A. Webson, E. Raff, and C. Raffel. 2022. [Crosslingual generalization through multitask finetuning](#).
- N. Thorer Sivesind and A. Bentzen Winje. 2023. [Turning poachers into gamekeepers: Detecting machine-generated text in academia using large language models](#).
- I. Solaiman, M. Brundage, J. Clark, A. Askel, A. Herbert-Voss, J. Wu, A. Radford, G. Krueger, J. Wook Kim, S. Kreps, M. McCain, A. Newhouse, J. Blazakis, K. McGuffie, and J. Wang. 2019. [Release strategies and the social impacts of language models](#).
- Y. Wang, J. Mansurov, P. Ivanov, J. Su, A. Shelmanov, A. Tsvigun, O. Mohammed Afzal, T. Mahmoud, G. Puccetti, T. Arnold, C. Whitehouse, A. Fikri Ajiand, N. Habash, I. Gurevych, and P. Nakov. 2024a. [Semeval-2024 task 8: Multidomain, multimodel and multilingual machine-generated text detection](#).
- Y. Wang, J. Mansurov, P. Ivanov, J. Su, A. Shelmanov, A. Tsvigun, C. Whitehouse, O. Mohammed Afzal, T. Mahmoud, T. Sasaki, T. Arnold, A. Fikri Aji, N. Habash, I. Gurevych, and P. Nakov. 2024b. [M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection](#). *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics Volume 1: Long Papers*, 1:1369–1407.

Source/domain/models excluded from the training	F1-macro	F1-micro	Accuracy
-	0.770	0.777	0.777
arXiv	0.790	0.795	0.795
PeerRead	0.829	0.831	0.831
Reddit	0.778	0.784	0.784
Wikihow	0.743	0.749	0.749
Wikipedia	0.726	0.736	0.736
ChatGPT	0.741	0.749	0.749
Cohere	0.782	0.788	0.788
Davinci-003	0.753	0.761	0.761
Dolly-v2	0.701	0.718	0.718

Table 3: Result metrics for BERT-base models

Source/domain/models excluded from the training	F1-macro	F1-micro	Accuracy
-	0.680	0.707	0.707
arXiv	0.550	0.615	0.615
PeerRead	0.855	0.858	0.858
Reddit	0.821	0.826	0.826
Wikihow	0.633	0.671	0.671
Wikipedia	0.741	0.756	0.756
ChatGPT	0.696	0.719	0.719
Cohere	0.657	0.689	0.689
Davinci-003	0.601	0.648	0.648
Dolly-v2	0.629	0.669	0.669

Table 4: Result metrics for RoBERTa-base-case models

Source/domain/models excluded from the training	F1-macro	F1-micro	Accuracy
RoBERTa base OpenAI detector	0.366	0.493	0.493
RoBERTa-mixed-detector	0.342	0.342	0.342

Table 5: Result metrics for RoBERTa base OpenAI detector and RoBERTa-mixed-detector models

Overfitting in Artificial Text Detection: Expanding on the SemEval-2024 Workshop Task 8

Sindre Bjørndal-Riis

University of Oslo

sindrsubj@uio.no

Abstract

In this paper we look at the tendencies of overfitting when doing artificial text detection, with a basis in the SemEval-2024 workshop task 8. The model trained for this paper uses many of the same techniques as those used in the workshop, and should thus transfer well to those models when looking at the case of overfitting.

The datasets used in the paper is 1) the same dataset as that used in the workshop, and 2) an unseen dataset, used to see whether the data is varied enough for the models to function well across domains.

The results of the experimentation show that any overfitting of the models is probably not caused by the dataset being too limited, but rather due to the training of the model not stopping early enough or not utilising regularisation. However, the results does show some decrease in performance across domains, which might suggest some overfitting to the domains present in the workshop dataset.

1 Introduction

Artificial text generation is becoming more and more available to the general public. This leads to concerns about people passing off texts written by artificial intelligence (AI) as their own, which can be problematic in several domains, and is of great concern in academic circles, where there is a desire to maintain academic integrity and ensure that the research being done is correct and verifiable. To combat this trend several researchers have been working on making models that can distinguish between artificially generated and human-written text in recent years.

The task of artificial text detection was given for the SemEval-2024 workshop, which is an international natural language processing research workshop. In the paper outlining the results from the workshop, [Wang et al. \(2024a\)](#) reflects that the models might generalise poorly across domains

and achieve worse results in those cases. Whether this concern is warranted is what this paper will be investigating.

In the workshop, the task of artificial text detection consisted of several subtasks, but the one that will be considered in this paper is subtask A, which focused on making a binary classifier that could detect whether a text is written by a human or generated by artificial intelligence. Subtask A was further separated into a monolingual part, which only looked at English texts, and a multilingual part, which included texts from many languages. This paper will only focus on the monolingual part of the task. The other subtasks were somewhat similar to subtask A, but looked at respectively which model had generated the artificial texts, and where the boundary between human and artificial text occurred in texts where there is a combination of both. The approaches used for subtasks A and B were fairly similar, and the results from this paper could give an indication as to whether the models from subtask B tend to generalise poorly across domains as well. The approaches used for subtask C differed somewhat to those of the other subtasks, and differ significantly from the approach used in this paper.

One potential risk when training machine learning models is that they are too specialised to the data that they are trained on. This paper will look into how robust a model trained on the dataset provided for the workshop is when tested on unseen datasets.

The model used for the discussion in this paper is a fine-tuned RoBERTa model. It is trained and evaluated both on the dataset that was used for the workshop, but is also evaluated on an unseen dataset consisting of abstracts from articles about coronaviruses, both human-written and artificially generated.

When training the model on the dataset used in the workshop and testing on the unseen dataset, the predictions achieves an accuracy of just above 0.5,

which is what we would expect when using a model without fine-tuning it. When training the model on the unseen dataset and testing on the dataset used in the workshop, the results are even worse, achieving an accuracy of less than 0.5. These results suggest that the models from the workshop do not tend to overfit to the dataset provided for the task, and that the dataset itself is still fairly varied and probably not the reason for any potential overfitting.

2 Background

Roelofs et al. (2019) conducted an experiment to analyse whether there is a tendency of overfitting in machine learning competitions, based on 120 competitions from the Kaggle platform. Their hypothesis was that an overuse of test sets when training models could lead to overfitting. The study found that there was no significant evidence of overfitting for the models trained in these competitions. This might be an indication that the models developed for the SemEval task will not be prone to overfitting either.

Other articles have found that networks trained to detect artificially generated texts tend to overfit to the topics they are trained on, and do not work well across domains (Su et al., 2023). Mitchell et al. (2023) used a zero-shot model for evaluation, and found that the model they trained worked well with predicting artificial texts amongst news texts, but the results were significantly worse for predicting artificial scientific texts, and it failed totally for German texts.

Ying (2019) researched solutions to overfitting. They divided these into three categories. The first is early stopping, which is stopping the training before it optimises. The second is reducing noises in the dataset, so that the models learn from representative data as much as possible. This is done by removing unhelpful or irrelevant data from the dataset by either removing complex rules or cutting down on the conditions in complex rules. The third is expanding the data so that the models have more data to train on. This could be done by combining datasets, or artificially expanding the dataset by changing the existing data slightly, depending on the task. The last strategy is regularisation, which involves selecting only useful features and not taking every feature into consideration. This is achieved by limiting useless features by trying to minimise a cost function, where a regularizer is added. Depending on the strategy, the function

could prioritise minimising small weights, or focus equally on making all the weights smaller regardless of original size.

One paper from earlier this year (Wang et al., 2024b) found that generalisation is difficult across domains, and that there is a lot of room for improvement. The models they looked at tended to classify machine-generated texts as written by humans. What they included, in contrast to previous work (Guo et al., 2023; Shamardina et al., 2022; Zellers et al., 2019), is a bigger variation of languages, domains and models. Previously it has been common to only use one or two languages, one model and one domain when training for artificial text detection, which might cause more overfitting than would otherwise be the case.

3 Setup

3.1 Model

One concern when training on a specialised dataset is that the model will overfit to that dataset or that domain, and achieve poorer results when testing on an unseen dataset. To investigate whether that is the case for this task, we need to have a model to test the datasets on. This model will be used with two different datasets, both the one presented in the workshop and an unseen one for this task.

Many of the top-performing models developed for subtask A in the workshop (Wang et al., 2024a) used fine-tuning and two of the top three models were fine-tuned on top of a pretrained RoBERTa model. Because of this, the model that will be used as a baseline for this experimentation will be a fine-tuned RoBERTa model, although with fewer additional features than most of the participating teams used. The model is trained on the training data from the workshop and evaluated on the development data for each epoch before testing it on the test data for the final accuracy score. For consistency across results, the hyperparameters for the model are not adjusted between any of the tests. The only modification done for a few of the tests is freezing the weights for all other layers than the head to see if full fine-tuning is necessary for these models and how it influences the effectiveness of the model with regards to cross-domain generalisation.

This does mean that the model used for this paper is not exactly the same as any of the final models created for the workshop. Considering that the underlying model and training data is the same as was

Split	Source	davinci-003	ChatGPT	Cohere	Dolly-v2	BLOOMz	GPT-4	Machine	Human
Train	Wikipedia	3,000	2,995	2,336	2,702	-	-	11,003	14,497
	Wikihow	3,000	3,000	3,000	3,000	-	-	12,000	15,499
	Reddit	3,000	3,000	3,000	3,000	-	-	12,000	15,500
	arXiv	2,999	3,000	3,000	3,000	-	-	11,999	15,498
	PeerRead	2,344	2,344	2,344	2,344	-	-	9,374	2,357
Dev	Wikipedia	-	-	-	-	500	-	500	500
	Wikihow	-	-	-	-	500	-	500	500
	Reddit	-	-	-	-	500	-	500	500
	arXiv	-	-	-	-	500	-	500	500
	PeerRead	-	-	-	-	500	-	500	500
Test	OutFox	3,000	3,000	3,000	3,000	3,000	3,000	18,000	16,272

Table 1: Dataset splits for subtask A in the SemEval task.

used for many of the contributions in the workshop, the model used in this paper should give an indication for how similar models from the workshop would evaluate as well.

3.2 Dataset

The dataset used for the task during the workshop consists of the splits seen in table 1. Since this is the dataset used during the workshop, it will also be used as a baseline for this paper, to make comparison as similar as possible. The dataset consists of texts from Wikipedia, Wikihow, Reddit, arXiv and PeerRead. arXiv and PeerRead contain academic articles and reviews of these. Wikipedia and Wikihow contain articles not aimed at academics, that anyone can edit. While most of these sites contain articles, one is a social media platform, Reddit, and contains posts written by the users of the page, which can be expected to contain less formal language than the data from the other sources. However, this is still a dataset which focuses heavily on more formal writing, with only 20% of the data potentially containing more informal language, dialects and slang.

To analyse how well the model works across domains, the publicly available ai-ga (Theocharopoulos et al., 2023) dataset is used in addition to the dataset provided for the workshop. This dataset consists of titles and abstracts from articles, where some of the abstracts are AI generated, and the rest are human-written. For the purposes of this paper, only the abstracts will be used for training and evaluating the model. The human-written abstracts are collected from the COVID-19 Open Research Dataset (CORD-19) (Wang et al., 2020), which is an aggregation of several papers from multiple sources that research Covid-19 and related viruses. To produce the AI-generated part of the dataset, Theocharopoulos et al. (2023) used GPT-3, giving it the prompt 'Create an abstract for a scientific journal with a formal tone, academic language, and a background story of the topic in a unique para-

graph with the title: \hat{t} ' where \hat{t} is the title from one of the articles in the original CORD-19 dataset. The final dataset with both human-written and AI generated abstracts consists of 28 662 entries where half are written by humans and half are generated by GPT-3. This is similar in size to the workshop test set. When training the model on the ai-ga dataset, it is further split into training, validation, and test sets.

The ai-ga dataset is very academically focused, and does, similarly to the workshop dataset, contain very formal language. However, it is unlikely that many of the articles in the workshop dataset covers exactly the theme presented in the ai-ga dataset, and so it should function as a way to analyse overfitting and the robustness of the dataset from the workshop.

In both datasets, every element also has a corresponding label that indicates whether it is written by a human (0) or machine (1), which are used for binary classification.

4 Results

To investigate whether the models could be sensitive to overfitting, they were run with several combinations of training and testing data, the results of which can be seen in table 2 and figure 1.

If every element in the test data had been assigned a random label (0 or 1) indicating which elements were predicted to be written by humans and which were predicted to be generated by a machine, we would assume an accuracy of around 0.5, the equivalent of flipping a coin. However, since we are basing our classifier on a pretrained model, the experimentation started by investigating how accurate predictions it yielded without training on the datasets at all. The results of this can be seen in table 2. From this we can see that the base assumption of 0.5 accuracy is fairly correct. Therefore, the models should achieve a higher accuracy score than 0.5 for us to be relatively certain that they have learned something from the training data rather than the models assigning labels by random guessing.

4.1 Training on the workshop dataset

To get a baseline the model was run with train, development and test data from the datasets used during the workshop and described in table 1. As can be seen from table 2, the model using the workshop dataset for both training and testing in this

Training data	Testing data	Accuracy
None	Workshop dataset	0.525
None	Ai-ga dataset	0.418
None	Smaller test ai-ga dataset	0.575
Workshop dataset	Workshop dataset	0.707
Workshop dataset	Ai-ga dataset	0.578
Frozen workshop dataset	Workshop dataset	0.721
Frozen workshop dataset	Ai-ga dataset	0.529
Ai-ga dataset	Workshop dataset	0.484
Ai-ga dataset	Ai-ga dataset	1.000
Frozen ai-ga dataset	Workshop dataset	0.478
Frozen ai-ga dataset	Ai-ga dataset	0.914

Table 2: Accuracy for the different training and test datasets with both frozen and unfrozen weights.

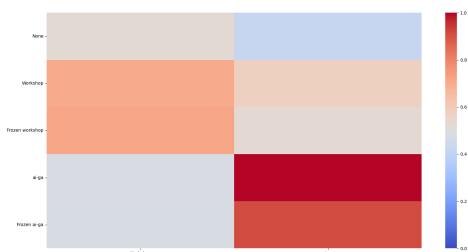


Figure 1: Heatmap of accuracies. The x-axis shows the dataset used for testing, and the y-axis the dataset used for training.

paper performed somewhat worse than the baseline from Wang et al. (2024a), which got an accuracy score of 0.885.

Using the same parameters, the model was then run using the full ai-ga dataset as the test set, but still using the training and development data from the workshop dataset. When comparing to the previous results, we can see that this scores lower. A different dataset having a lower accuracy is to be expected, but this being higher than when using the model without fine-tuning is promising for the model not overfitting.

In addition, the same two tests were run with frozen weights, where the training only optimised the head layers. Freezing layers is a way of combating overfitting as it means that not all layers are optimised to the training dataset. If the results for the unseen dataset becomes noticeably higher or the results of the workshop dataset becomes significantly lower when using frozen layers, it could indicate that the original model tends to overfit.

From these tests we can see that freezing the weights did not have a significant effect on either dataset combination. However, it achieved a slightly higher result with the workshop dataset. This further indicates that the model does not tend

to overfit to the training dataset. It could also suggest that it is good enough to only train the head layers, which significantly decreases the required training time, although it might then get less general results, as the frozen weights achieved a slightly worse result when testing on the ai-ga dataset. When freezing the weights, the ai-ga dataset only just got more than 0.5 accuracy, which is still higher than what we achieved without fine-tuning, but not by much.

4.2 Training on the ai-ga dataset

To investigate how well the baseline model performs, it is also interesting to see how well the model performs if we train it on the ai-ga dataset and test it on respectively the ai-ga and workshop datasets. The results of these experiments can give an indication as to whether the workshop dataset is varied enough to use for training a general classifier across multiple domains or if it should be modified somewhat for cross-domain use. If the result of using the datasets this way is worse than when training on the workshop dataset, it might indicate that the original dataset is good enough to train a model that generalises adequately for cross-domain tasks. If it does not, it indicates that the workshop dataset does not generalise well, considering that the ai-ga dataset is very specialised in regards to domain, as well as being fairly small.

It is in this experiment important to note that the aggregated workshop training, validation and test datasets are significantly larger than the ai-ga dataset. This means that we can assume some decrease in performance. Additionally, the ai-ga dataset is a lot less varied with regards to topics in the texts and the type of texts included, which would further the assumption of a decrease in performance. Although this might seem problematic, for this experimentation it is helpful for giving an indication as to how robust the workshop dataset is, since the ai-ga dataset is exactly what we do not want to see in training data for a general classifier.

Since the ai-ga dataset is not already split into training, validation and testing datasets, it needed to be split into those sections before training. The dataset was split into a 60% / 20% / 20% split, consisting of respectively 17 197, 5 732, and 5 732 samples. In comparison to the workshop dataset the validation set is fairly similar in size, while the other two are noticeably smaller. When fine-tuning all the weights and testing on the ai-ga dataset the model achieved an accuracy of 1.000, which is

suspiciously high. This indicates either that the model overfits to the training data or that the dataset is not varied enough. When freezing the weights, the accuracy drops a little, down to 0.914. Similarly to what we saw in the previous section, this is not a significant decrease in performance, and could suggest that the model does not in fact overfit to the dataset, but rather that the dataset is not varied enough to have many unseen instances.

When testing on the workshop test dataset, the same data was used for the training and validation sets as when testing with only the ai-ga dataset, that is respectively 60% and 20% of the full ai-ga dataset. The workshop test set achieved a much worse accuracy score than was seen when using only the ai-ga dataset. When training on the ai-ga dataset, the workshop test set got a slightly worse accuracy for the frozen weights compared to training all of the layers, but not significantly so.

The fact that the workshop dataset gets a noticeably worse score when training on the ai-ga dataset, and even worse than the other way around or with no fine-tuning, suggest that this training combination tends to overfit to the training data more than using the workshop dataset for training, which indicates that the workshop dataset itself does not pose a significant problem with regards to overfitting.

5 Discussion

Giving a definite conclusion as to whether the model trained on the workshop dataset is prone to overfitting or not is slightly difficult. On the one hand, when testing on the unseen dataset, the accuracy score is not much above 0.5. On the other hand, it is noticeably better than when testing with the ai-ga dataset on the RoBERTa model that is not fine-tuned. However, since the distance between the accuracy score for the baseline model tested on the baseline test set (0.707) and the baseline model tested on the ai-ga dataset (0.578) is fairly small, it is not very likely that it overfits.

What is interesting to look at when investigating overfitting in machine learning, is whether it is the dataset or the model that seems to be the root of the problem. One reason for overfitting can be that the dataset is not varied enough, and makes it so that the model does not learn general enough features to translate well to other domains. Another reason can be that the model fine-tunes for too long so that even if the dataset is in theory varied enough to avoid overfitting, the model focuses too much

weight on random details that happen to be in the training set rather than general features that are relevant for the task.

From the results it does not seem that the dataset given during the workshop is the main problem in this case. Switching the datasets with regards to which are used for training and which are used for test shows that training on the ai-ga dataset yields noticeably worse results on the workshop dataset than the other way around.

The training of the model could of course also be a problem, and it might look like that from the results of the model trained on the ai-ga dataset, but then a higher accuracy score would be expected for the model trained on the workshop dataset as well. Seeing as the model trained on the workshop dataset achieves a lower accuracy score than the baseline in the workshop, and not a lot higher than the model tested on the ai-ga dataset, it does not seem that the training of the model is a problem either. There was also made efforts in assuring that the model is not the problem in this case, such as having fairly few epochs during training, so that it will stop before it achieves the 'optimal' weights.

As discussed in [Mitchell et al. \(2023\)](#), it is not uncommon for networks trained to detect artificially generated texts to generalise poorly across domains, which seems to be the case for this task as well. One possible reason posed by [Wang et al. \(2024a\)](#) was that all of the models trained for this task are black-box models, whereas white-box models¹ tend to show a greater sign of robustness with regards to cross-domain utilisation. However, it does not appear that the reason for the poor cross-domain generalisation in this case is overfitting to the training data.

It is worth noting that it is possible that the models used in the workshop tend less toward overfitting than what is seen in this article, as there are many different methods for training a model used in the workshop and many variations on what has been done in this paper, which might adjust for the problems discussed. However, the model used for the experimentation in this paper is based on what most of the top-performing models in the workshop used and should give some indication as to how they perform with cross-domain tasks.

¹Models which have results that can be explained, e.g. using patterns or rules ([Loyola-González, 2019](#)).

6 Future work

In the workshop several subtasks were presented, but this article only focused on the first of these subtasks. The subtasks presented are all focused on detecting artificial texts, so the results from this paper could be an indication of how the models for the other subtasks perform as well in regards to cross-domain utilisation. However, there needs to be done further research into those tasks to determine whether that is the case.

Researching how models using different strategies than fine-tuning compare could also be interesting, as they might have different results with regards to overfitting than what can be seen for the models using fine-tuning.

7 Limitations

One of the main limitations of this article is that the tests were only run on one unseen dataset, which contains similar texts to parts of the original dataset. This means that the resulting model has seen at least some similar texts before, and seeing how the model performs with texts from totally unseen domains could make for an interesting comparison. Especially using a dataset with no articles, but rather more informal languages, for instance with conversations could be interesting. Doing so could expand the results, giving a more thorough understanding of how extensive the problem of overfitting is in artificial text detection.

Additionally, the unseen dataset used is fairly small, especially when split into training, development and test splits. Using a larger dataset, of similar size to the workshop dataset, to see how that holds up in comparison to the original dataset might give a fairer comparison, and thus more conclusive results.

8 Conclusion

In this paper we have looked at the tendencies of overfitting when doing artificial text detection, with a basis in the SemEval-2024 workshop task 8 subtask A. The model trained for this paper uses the same techniques as several of the models used in the workshop, and should thus transfer well to those models when looking at the case of overfitting. Specifically the model used for this paper uses RoBERTa as the underlying model and is finetuned on the datasets described in section 3.2.

The datasets used for the experimentation in this paper were 1) the same dataset with the same splits

as in the workshop task, and 2) a different, unseen dataset based on abstracts from papers about coronaviruses. The second dataset was used to see whether the data provided in the workshop is varied enough for the models to generalise well across domains.

The results show that the classifier trained on the workshop dataset achieves a much higher accuracy when tested on the baseline test set, than it does when tested on the ai-ga dataset. Similar results can be seen when the classifier is trained on the ai-ga dataset, but tested on the baseline test set. However, training on the workshop dataset does achieve slightly higher accuracies for the unseen data than using the classifier without fine-tuning. This is not the case when training on the ai-ga dataset, which indicates that the dataset itself is fairly varied, and any overfitting tendencies from the models presented in the workshop is due to the models not stopping training early enough or not utilising regularisation.

All of this shows no tendency from the model towards overfitting to the data. However, the model still does not translate well across domains.

Acknowledgments

I would like to thank Vladislav Mikhailov for presenting this track for the exam as well as giving guidance during the home exam. Additionally, I would like to thank all the reviewers who has given me helpful feedback on how to present this paper in a more thorough and understandable way.

References

- Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. [How close is chatgpt to human experts? comparison corpus, evaluation, and detection](#). *Preprint*, arXiv:2301.07597.
- Octavio Loyola-González. 2019. [Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view](#). *IEEE Access*, 7:154096–154113.
- Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. [Detectgpt: Zero-shot machine-generated text detection using probability curvature](#). *Preprint*, arXiv:2301.11305.
- Rebecca Roelofs, Vaishaal Shankar, Benjamin Recht, Sara Fridovich-Keil, Moritz Hardt, John Miller, and Ludwig Schmidt. 2019. [A meta-analysis of overfitting in machine learning](#). In *Advances in Neural*

Information Processing Systems, volume 32. Curran Associates, Inc.

news. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Tatiana Shamardina, Vladislav Mikhailov, Daniil Chernianskii, Alena Fenogenova, Marat Saidov, Anastasiya Valeeva, Tatiana Shavrina, Ivan Smurov, Elena Tutubalina, and Ekaterina Artemova. 2022. [Findings of the the ruatd shared task 2022 on artificial text detection in russian](#). In *Computational Linguistics and Intellectual Technologies*. RSUH.

Jinyan Su, Terry Yue Zhuo, Di Wang, and Preslav Nakov. 2023. [Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text](#). *Preprint*, arXiv:2306.05540.

Panagiotis C. Theocharopoulos, Panagiotis Anagnostou, Anastasia Tsoukala, Spiros V. Georgakopoulos, Sotiris K. Tasoulis, and Vassilis P. Plagianakos. 2023. [Detection of fake generated scientific abstracts](#). In *2023 IEEE Ninth International Conference on Big Data Computing Service and Applications (Big-DataService)*, pages 33–39.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darin Eide, Kathryn Funk, Yannis Katsis, Rodney Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex Wade, Kuansan Wang, Nancy Xin Ru Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. [Cord-19: The covid-19 open research dataset](#). *Preprint*, arXiv:2004.10706.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Chenxi Whitehouse, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024a. [Semeval-2024 task 8: Multidomain, multimodel and multilingual machine-generated text detection](#).

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Toru Sasaki, Thomas Arnold, Alham Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024b. [M4: Multi-generator, multi-domain, and multilingual black-box machine-generated text detection](#). In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1369–1407, St. Julian's, Malta. Association for Computational Linguistics.

Xue Ying. 2019. [An overview of overfitting and its solutions](#). *Journal of Physics: Conference Series*, 1168(2):022022.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. [Defending against neural fake](#)

Using Statistical Features to Improve the Performance of Large Language Models in Artificial Text Detection

Andreas Poole

andrepoo@uio.no

Abstract

Artificial Text Detection (ATD) is the task of identifying whether a text has been generated by a machine or written by a human. This paper explores how this task can be performed by using Large Language Models (LLMs), statistical features (text statistics), and a combination of the two. There is evidence that machines and humans generate texts with different statistical properties and the main question asked in this paper is whether LLMs are able to fully comprehend these differences when used in ATD, or if statistical features can improve the performance of LLM-predictors. This study finds some support for the second case.

1 Introduction

Artificial Text Detection (ATD) is the task of identifying if a text has been generated by a machine or written by a human. The task is closely related to that of detecting misuse of LLMs, where machine generated texts are presented as written by humans, or vice versa. Cases of misuse have been broad, with examples including LLMs infiltrating internet discussion forums and public opinion web pages (Crothers et al., 2023).

Misuse, both real and potential, highlights the need for knowledge about ATD, which has sparked some activity among researchers. One of these activities is the academic research competition "SemEval-2024 Task 8: Multigenerator, Multidomain, and Multilingual Black-Box Machine-Generated Text Detection"¹ which provides the setting for this study.

2 The SemEval Competition

The competition was open and anyone could submit their contributions. It consisted of three subtasks, where submissions are made independently

to each subtask, and even tracks of subtasks. This paper simulates an artificial submission to Subtask A: "Human vs Machine classification". This is a binary classification task where the only goal is to predict whether a text was written by a human or generated by a machine. Machine generated is assigned the positive class, which makes human written the negative class.

The subtask consisted of two tracks: Monolingual and multilingual. The monolingual track consist of classification of only English texts, while the multilingual deals with classification of texts in several different languages.

A shared requirement for the task is to not use external datasets, i.e. it is not allowed to use other data than the data provided with tasks to train models entering the competition. This requirement has been followed in this study.

More details about the competition can be found in the SemEval Shared Task Paper (Wang et al., 2024a).

2.1 Data

The dataset provided for the competition was an extension of the M4 dataset (Wang et al., 2024b). The dataset consists of samples from different sources, e.g., web domains like Wikipedia, which have both human written and machine generated texts, generated by different generators, i.e., LLMs.

In the competition development phase, the participants were provided with training sets and development sets. For the monolingual track the training set consisted of 119,757 samples of which 52.9% was human written and 47.1% was machine generated. For the multilingual track the training set consisted of 172,417 samples. As in the monolingual training set the samples was fairly balanced between human and machine generated texts. The dataset was heavily skewed towards English. 79.2% of the samples were English texts, while the remaining 20.8% was split between four different

¹<https://github.com/mbzuai-nlp/SemEval2024-task8>

Split	Source	davinci-003	ChatGPT	Cohere	Dolly-v2	BLOOMz	GPT-4	Machine	Human
Train	Wikipedia	3,000	2,995	2,336	2,702	-	-	11,033	14,497
	Wikihow	3,000	3,000	3,000	3,000	-	-	12,000	15,499
	Reddit	3,000	3,000	3,000	3,000	-	-	12,000	15,500
	arXiv	2,999	3,000	3,000	3,000	-	-	11,999	15,498
	PeerRead	2,344	2,344	2,342	2,344	-	-	9,374	2,357
Dev	Wikipedia	-	-	-	-	500	-	500	500
	Wikihow	-	-	-	-	500	-	500	500
	Reddit	-	-	-	-	500	-	500	500
	arXiv	-	-	-	-	500	-	500	500
	PeerRead	-	-	-	-	500	-	500	500
Test	Outfox	3,000	3,000	3,000	3,000	3,000	3,000	18,000	16,272

Table 1: Subtasks A: Monolingual Binary Classification. Data statistics over Train/Dev/Test splits (Reproduced from Wang et al., 2024a, Table 1)

languages.

The development sets were blind datasets (i.e. no gold labels). However, even with the gold labels hidden, the organizers still gave the participants the opportunity to view their performance on the development sets during the development phase. This meant that they had some possibility to tune their models to perform better on the development set. After the submission deadline, the gold labels of the development set was released, along with the test set used to rank the participants.

The monolingual test set was from a single surprise source, Outerfox, not present in the training data, the data also included a new generator, GPT-4, in addition to the same generators as in the training set. The test set consisted of an equal amount of samples from each generator. Unfortunately, only the gold labels for the test set was released after the competition, the labels for the generators was not released.

The multilingual test set was again heavily skewed towards English texts, although to little less degree than the training set, with 66.7% of the texts being English. The test set included two new surprise languages, but due to the dominance of English there were few samples of these surprise languages.

The special usage of the development set makes it difficult to use in this "simulated" study, and for this reason it is not used in this study. Only the training set is utilized in the development phase and the test data is used to measure performance on the final models.

Data statistics for the monolingual track are shown in Table 1 and for the multilingual track in Table 2.

3 Research Questions

Studies, e.g., by Fröhling and Zubiaga (2021) and Adilazuarda (2023) show that statistical text fea-

Split	Language	davinci-003	ChatGPT	LLaMA2	Jais	Other	Machine	Human
Train	English	11,999	11,995	-	-	35,036	59,030	62,994
	Chinese	2,964	2,970	-	-	-	5,934	6,000
	Urdu	-	2,899	-	-	-	2,899	3,000
	Bulgarian	3,000	3,000	-	-	-	6,000	6,000
	Indonesian	-	3,000	-	-	-	3,000	3,000
Dev	Russian	500	500	-	-	-	1,000	1,000
	Arabic	-	500	-	-	-	500	500
	German	-	500	-	-	-	500	500
	Italian	-	-	3,000	-	-	3,000	3,000
Test	English	3,000	3,000	-	-	9,000	15,000	13,200
	Arabic	-	1,000	-	100	-	1,100	1,000
	German	-	3,000	-	-	-	3,000	3,000
	Italian	-	-	3,000	-	-	3,000	3,000

Table 2: Subtasks A: Multilingual Binary Classification. Data statistics over Train/Dev/Test splits (Others generators are Cohere, Dolly-v2 and BLOOMz) (Reproduced from Wang et al., 2024a, Table 2)

tures (features) show some predicting power in the ATD setting. LLM’s have also been widely applied to this task. It should be obvious that when a LLM is trained or fine-tuned to the ATD task, it will learn at least some of the differences in the features and implicitly use them when making predictions. However, this does not necessarily mean that all information in the features are absorbed by an LLM. In our study we attempt to look at this effect by investigating if a combined model using both a LLM and features can benefit in the ATD task compared to LLM-only predictors. In addition we study if any benefit from features depends on the size of the LLM used, if smaller LMs benefit to a higher degree than larger LMs.

The main focus of the study is on the monolingual track, however we also attempt the multilingual track. Where the research question is how well the prediction power of features, which mostly is designed to use on English texts, transfers to other languages.

4 Predictor Models

4.1 LLM

LLMs, and specifically fine-tuning LLM, are the "go to candidate"-models for most NLP tasks these days. This is also a natural candidate for ATD, and was also chosen as a baseline in the SemEval-competition.

LLMs have the advantage that they work very well over a broad spectrum of tasks, but at the cost of being expensive when it comes to computing resources and also score low on interpretability.

The selected LLMs for this study were the same as the baseline models used in the competition: RoBERTa (Liu et al., 2019) for the monolingual track and XLM-RoBERTa (Conneau et al., 2020)

for the multilingual track. To investigate how model size influence results, we used RoBERTa with two different size configurations in the monolingual track, base² with 125 million parameters and large³ with 355 million parameters. For the multilingual track only the base XLM-RoBERTa⁴ model was used.

The fine-tuning of the LLMs was conducted in a similar way to how it was done in the organizers baseline code⁵. The hyperparameters used for fine-tuning the LLMs were:

- *Train-Validation set split*: 90%-10%
- *Number of epochs*: 5, early stopping with patience of 2
- *Batch size*: base models: 32, large model: 16
- *Learning rate*: $2 \cdot 10^{-5}$
- *Weight decay*: 0.01

The validation set was used to decide early stopping during training.

4.2 Feature based

The feature based approach, sometimes also called stylometric attribution (e.g., by Uchendu et al. (2023)) to text analysis is about calculating statistical measurements on texts and using these measurements to infer properties of the texts. This can be done at character level, word level, n-gram level or text level, depending on how the texts are sub-sampled when calculating the statistics. There are several such features which can be measured on text. An extensive survey can be found in Lagutina et al..

In the context of ATD: "The feature-based approach to discriminate between human and machine text is grounded on the assumption that there are certain dimensions in which both types differ." (Fröhling and Zubiaga, 2021).

There are several reasons one may hypothesize that humans and machines may generate texts which may, on average, may score differently on such metrics.

²<https://huggingface.co/FacebookAI/roberta-base>

³<https://huggingface.co/FacebookAI/roberta-large>

⁴<https://huggingface.co/FacebookAI/xlm-roberta-base>

⁵<https://github.com/mbzuai-nlp/SemEval2024-task8/tree/main/subtaskA/baseline>

Zipf (1949) connected human behavior to texts written by humans through the principle of least effort. This might suggest that humans tend to use shorter words, write shorter sentences and use fewer stop words than machines, simply for the case of minimizing effort.

Further; Gehrmann et al. (2019) "describe that language models fail to use synonyms and references as humans do, but rather stick to the repetition of the same expressions, leading to a lack of syntactic and lexical diversity in machine text." (as cited in Fröhling and Zubiaga, 2021). Which might lead to higher Vocabulary Richness in human texts.

In this study the selected features were mostly based on the features described in Adilazuarda (2023), with the exception of stop word frequency and maximum sentence length which was also included in this study.

The features used in this study were:

- *Vocabulary Richness*: Rate of number of unique words to total number of words (aka. type-token ratio for word tokens)
- *Average Word Length*: Average length of words measured in characters
- *Average Sentence Length*: Average length of sentences, measured in number of words (signs removed)
- *Maximum Sentence Length*: Maximum length of sentence in text, measured in number of words (signs removed)
- *Stop Word Frequency*: Rate of occurrences of stop words to total number of words.

The Natural Language Toolkit, nltk⁶ by Bird et al. (2009), was used in the calculation of these features. Word tokenizer was used to separate words, sentence tokenizer to separate sentences, and the English stop words list were utilized when calculating stop word frequency.

The feature based classifier was a simple logistic regression classifier⁷. The values for the hyperparameters used by this classifier were:

- *Regularization parameter*: 1
- *Penalty*: ℓ_2

⁶<https://www.nltk.org/>

⁷https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

- *Solver*: lbfgs

In many ways, the feature based approach has opposite properties to that of the LLM approach. Features are cheap and fast to compute, and also are highly interpretable. Unfortunately they often perform poorly. Fröhling and Zubiaga (2021) compares their performance with Solaiman et al. (2019) and finds that for their feature based predictor "our best ensemble model lagging behind 18 percentage-points in accuracy" for the largest test set used by both authors.

4.2.1 Multilingual Track

Many of these features are designed with English and similar languages in mind and might not transfer well to other languages. English stop words, does of course not transfer to any other languages. Other features should transfer well to other languages using alphabetic writing systems, but most likely do not transfer well to other languages using logograms or syllabaries, like Chinese.

4.3 Combined Model

In the combined model the probabilities created by the LLM were converted to logits. The logits were used in addition to the text features as input to a logistic regression classifier. The values for the hyperparameters for this classifier was the same as described in 4.2. The general idea here is that the features might be able to help the LLM in samples where the LLM is in doubt, i.e. the probability assigned to the predicted class is low (close to 0.5 in the binary case), and the features might just push the predictions from the wrong to the correct class (more often then it does the opposite).

5 Data Analysis

A nice feature with features is that they are highly interpretable, and are easy to use to get some insight into the dataset. Figures 1 to 5 shows the densities of values for the selected features for the samples in the monolingual training set.

From these visualizations it can be observed some trends in differences between feature statistics for human written and machine generated texts, in the training set. Some of the features statistics are in line with the hypotheses and citations presented in 4.2. Human written texts do on average have shorter words, lower stop word frequency and higher vocabulary richness than machine generated texts. However, human written texts tend to have

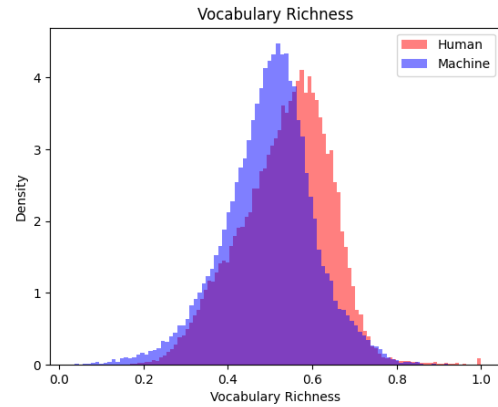


Figure 1: Vocabulary Richness

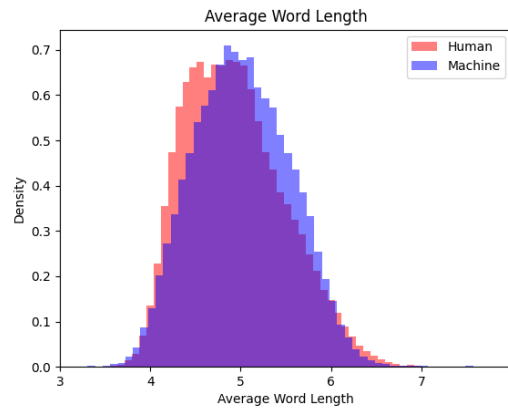


Figure 2: Average Word Length

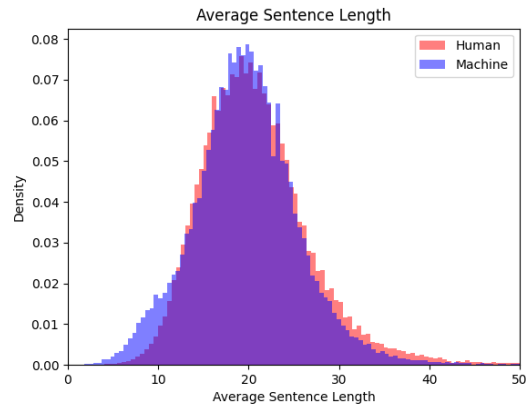


Figure 3: Average Sentence Length

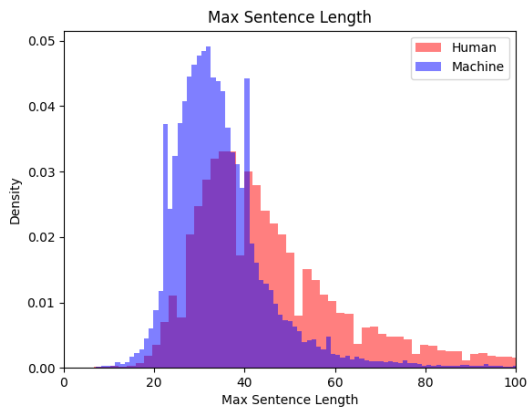


Figure 4: Maximum Sentence Length

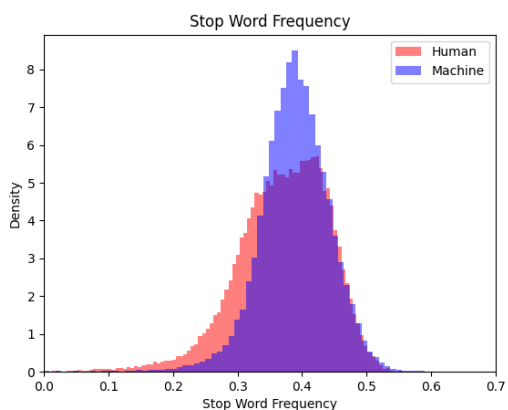


Figure 5: Stop Word Frequency

Metric	LLM	Features	LLM+Features
Accuracy	82.11%	65.55%	85.51%
Precision	77.38%	64.64%	83.42%
Recall	93.17%	75.94%	90.37%
F1-Score	84.55%	69.84%	86.75%

Table 3: Performance metrics, monolingual track, RoBERTa-base LLM-only, features-only and LLM+Features

longer sentences than machine generated texts, and specifically tends to have higher maximum sentence lengths.

6 Results

6.1 Monolingual Track

The performance metrics for RoBERTa-base, features based model and combined model using this LLM are shown in Table 3.

Not surprisingly the features-only model score poorly, and a LLM alone scores substantially better. For both models, recall is higher than precision. This means that the models tend to predict that texts are machine generated in too many cases, at the cost of many false positives. This is not an effect of tending to predict to the majority class, as machine is actually the minority class in the training set.

The results from the participants in the SemEval competition showed a similar pattern, with many participants posting recalls in the 90-ties, but very few posting similarly strong values for precision (Wang et al., 2024a, Table 9).

One can be tempted to assume this effect is caused by the introduction of the surprise source Outerfox. That the new source represents a distributional shift, fooling the models to believe many of the human written texts in the test set are generated by machines, but the effect is even stronger in the multilingual track, and in that case in can not be attributed to the introduction of this source.

For accuracy, the results in our study are in line with previous research. For the features alone the results are in the same range as those achieved by (Adilazuarda, 2023, Table 3). This study uses similar features as our study. The results for fine-tuned LLMs in the same study is also similar. As shown in (Fröhling and Zubiaga, 2021, Table 1) their study achieves somewhat higher scores for their feature approach, on average in the high 70-ties, but their study uses more complex features than our study.

Metric	LLM	LLM+Features
Accuracy	76.04%	84.54%
Precision	69.91%	82.01%
Recall	95.47%	90.39%
F1-Score	80.71%	86.00%

Table 4: Performance metrics, monolingual track, RoBERTa-large LLM-only and LLM+Features

For the combined model, the problem of over-predicting to the positive class is less prominent. The model predicts the machine class in less cases and gives more balanced predictions. This gives a lower recall, but higher scores overall. This happens even if the feature based model has the same problem with over-predicting to the machine class. What we may see here is that the features in some way regularize the LLM-predictions towards predictions to the majority class, which is human written texts. This gives an increase in false negative predictions, but the decrease in false positives is higher leading to higher overall scores.

The performance metrics for RoBERTa-large and combined model using this LLM are shown in Table 4.

Using a larger LLM yields yet more surprising results. When fine-tuned in a similar fashions, the larger LLM scores worse than the smaller LLM. The effect of over predicting to the machine class is even stronger in this case, dropping precision to a very low level and along with it also the overall scores. The combined model scores worse than the one using RoBERTa-base, but the outperformance compared with the LLM-only predictor is higher. It’s regularizing effect has an even stronger effect. The decrease in recall is larger than for the smaller LLM, but a significant increase in precision more than weight up for this. In a way this illustrates how our combined model profits from poor behavior of the LLM.

This result makes it difficult to draw any conclusions about whether the benefit from features depends on the size of the LLM. Such insight would require the larger LM to perform better than the smaller LM.

6.2 Multilingual Track

The performance metrics for XLM-RoBERTa-base, features based model and combined model using this LLM are shown in Table 5.

For the LLM the performance is a little worse than for the monolingual case. Again, we observe

Metric	LLM	Features	LLM+Features
Accuracy	80.48%	67.78%	86.42%
Precision	72.94%	65.04%	79.80%
Recall	99.60%	82.89%	99.07%
F1-Score	84.21%	72.89%	88.40%

Table 5: Performance metrics, multilingual track, XLM-RoBERTa-base LLM-only, features-only and LLM+Features

that recall is much higher than precision, and even higher than in the monolingual track.

Looking at the performance of the participants in the competition (Wang et al., 2024a, Table 10) we also see an even stronger trend towards high recall and low precision than in the monolingual track.

For the features the performance is just as good as in the monolingual track. Considering the dominance of English texts and that all languages in the test set use alphabetic writing systems (i.e., are similar languages), the results are not unreasonable.

The combined model shows similar improvement compared to the LLM-only predictor, as in the monolingual track. The regularization effect again seems successful, giving a boost to precision, but in this case with only a small drop in recall. This means that most of the changed predictions from the positive to the negative class was correct, leading to a substantial increase in overall performance on this particular test set.

For all models, it would have been interesting to study the variance in performance between the different languages, but unfortunately this is not feasible due to missing language labels for the samples in the test set.

7 Limitations of study

The study is limited in general, as it is quite one dimensional, in the sense that it explores little variations in the choices made when designing the experiments. The only variation in the experiments was to run the monolingual model for RoBERTa in two size configurations. The results are therefore also limited to the selected LLMs. To further validate the results across multiple language models, the study could have used multiple LLMs, including decoder only models like GPT.

The features used in the study is only a small subset of the features that can be relevant for ATD. In addition, the selected features are simple in nature and limited to word and sentence level. There

are several other features that could have been included.

One candidate is type-token ratio for n-grams, which could have measured repetition of word sequences, and supplemented the single word variant used in this study.

Another interesting feature is Perplexity, as it measures the improbability of a text, from the point of view of a LLM, and it could be hypothesized that a LLM would assign higher Perplexity to texts written by a human than to texts generated by a machine, thus making it relevant for ATD.

Some features used in this study can be connected to work by Zipf (1949). Further features that measure how well texts follow Zipf's Law, e.g., features that measure goodness of fit of a text's word length frequencies compared to expected frequencies according to Zipf's law, could be interesting, as this law is directly related to human behavior and as such can be relevant for ATD.

Another limitation of this study is that it has no feature selection, in the sense that all available features are fed into the logistic regression classifier. The inclusion of additional features would necessitate the inclusion of a proper feature selection method, where features that might have negative predictive power are not used when fitting the classifier.

In summary, there are several limitations to this study. These could have been expanded on to make it a more interesting study, with more robust results.

8 Conclusion

The results in this study show that combining LLMs with features can improve results in ATD, both in monolingual and multilingual settings. However, this result can be attributed to applying the method to a case where LLM-predictors show somewhat extreme behavior, which makes the conclusion depend on presence of this kind of behavior of LLMs. A more interesting question would be what would happen in a case where LLMs did not show this behavior? E.g. a case where LLM-predictors produced similar rates of false positives and false negatives. Most likely, the method would not perform as well in this case, but might there still be some benefits from combining features with LLMs, or will there be no or even a negative effect? This question will remain unanswered in our study, but may lay the ground for future work.

References

- Muhammad Farid Adilazuarda. 2023. [Beyond Turing: A Comparative Analysis of Approaches for Detecting Machine-Generated Text](#). *arXiv preprint arXiv:2311.12373*.
- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc.
- 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](#). *CoRR*, abs/1911.02116.
- Evan Crothers, Nathalie Japkowicz, and Herna Viktor. 2023. [Machine-generated Text: A Comprehensive Survey of Threat Models and Detection Methods](#). *IEEE Access*, PP.
- Leon Fröhling and Arkaitz Zubiaga. 2021. [Feature-based detection of automated language models: tackling GPT-2, GPT-3 and Grover](#).
- Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. [GLTR: Statistical Detection and Visualization of Generated Text](#). pages 111–116.
- Ksenia Lagutina, Nadezhda Lagutina, Elena Boychuk, Inna Vorontsova, Elena Shliakhtina, Olga Belyaeva, and Ilya Paramonov. [A Survey on Stylometric Text Features](#).
- 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.
- Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. 2019. [Release Strategies and the Social Impacts of Language Models](#).
- Adaku Uchendu, Thai Le, and Dongwon Lee. 2023. [Attribution and Obfuscation of Neural Text Authorship: A Data Mining Perspective](#).
- Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Chenxi Whitehouse, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024a. [SemEval-2024 Task 8: Multidomain, Multimodal and Multilingual Machine-Generated Text Detection](#).
- Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Toru Sasaki, Thomas Arnold, Alham Aji, Nizar

Habash, Iryna Gurevych, and Preslav Nakov. 2024b. [M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection](#). In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1369–1407, St. Julian's, Malta. Association for Computational Linguistics.

George Kingsley Zipf. 1949. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley.

Zero-shot Artificial Text Detection Using Instruct Fine-Tuned LLMs

Zoia Butenko
zoiab@uio.no

Taras Andrushko
tarasan@uio.no

Abstract

The rapid development of large language models (LLMs) has revolutionized various fields by enabling the generation of high-quality text, but it also raises concerns about the proliferation of machine-generated content and its detection. This paper investigates the effectiveness of instruction-tuned LLMs for zero-shot and few-shot detection of machine-generated text, a task that has gained importance in the context of increasing online content generated by AI. We evaluate the performance of these approaches in comparison to a fine-tuned model using data from the SemEval2024 Task 8. Our findings indicate that both zero-shot and few-shot setups with instruction-tuned LLMs perform poorly, exhibiting a significant bias towards labeling texts as human-written. In contrast, the fine-tuned model, optimized with LoRA and trained on a small subset of instances, demonstrates superior performance, although its effectiveness varies across different text domains. These results highlight the limitations of zero-shot and few-shot methods for this task and suggest that fine-tuning remains crucial for accurate detection of AI-generated text. Additionally, our study underscores the importance of considering domain-specific biases and other factors such as text length and the generator used in developing robust AI detection models.

1 Introduction

The rapid development of large language models in recent years has not only led to advanced chatbots facilitating many tasks but also to big amounts of artificially generated data that is nearly impossible to spot with an untrained eye. The implications of advanced generative technology in the fields of computer vision, sound, or textual data have led to complete mistrust of image or sound evidence by some while being extremely deceptive for others. Leveraging large language models for large-scale production of artificially generated texts

is now becoming a new industry, both profit- and politically-driven. On one hand, generated tasks can be used to fill in gaps in resources (for example, automatically translated pages), on the other hand, the deceptive nature of artificially generated textual data is ethically ambiguous.

Detectors for artificially generated text have become incredibly popular, for example, those implemented in university anti-plagiarism tools have been a subject of discussion since the release of ChatGPT made artificially generated texts as widespread as they are today. The misinterpretation of anti-plagiarism tools' results has repeatedly led to issues with employment and enrollment for those whose works have been mistakenly flagged as AI generated when the detector showed probabilities higher than a certain threshold. On the other hand, AI-generated content has flooded social media platforms and search engine results, a report by Europol estimating that as much as 90% of content online may be AI-generated by 2026 (Europol, 2022).

The SemEval2024 Task 8 introduced a shared task that focuses on the detection of Machine-Generated Text (MGT) (Wang et al., 2024a). Interestingly, the best systems in all categories used LLMs to achieve the result, although a plethora of approaches were present, including perplexity-based, probabilistic features, training neural networks from scratch, and fine-tuned models. In Subtask A (Monolingual Binary Classification), most teams employed LLMs either for feature extraction or fine-tuning. Zero-shot approaches, however, were less popular, with only 2 teams represented. In this work, we will attempt to prompt LLMs using zero- and few-shot techniques for binary classification. Prompting LLMs for any task, let alone a task of such complexity as artificial text detection, is incredibly tricky. The results vary significantly depending on the prompt formulation and generation parameters such as temperature. Instruction-tuned

large language models are a class of language models that have been fine-tuned on datasets consisting of natural language instructions paired with desired outputs, making it easier to prompt LLMs for most popular tasks (Zhang et al., 2023). During the SemEval2024 Task 8 instruction-tuned LLMs were not used by any of the teams, but could significantly improve performance in few-shot setups. We will thus try leveraging instruction-tuned models for few-shot approaches and report the performance comparison. We will also analyze the results in respect to domain-specific data to see whether few-shot setups with instruction-tuned LLMs could have bias in that regard.

2 Data, existing approaches, task formulation

2.1 Dataset

The monolingual binary classification subtask A dataset consists of 119,757 instances in the train subset and 5,000 instances in the development subset. Each instance is annotated for the label (0 for human-written and 1 for artificially generated), the model used to generate ('human' if the text is human-written), and the source of data. The overview of generative models represented in the dataset can be found in Figure 2, overview of sources of data is in Figure 3. Overall, the dataset is more or less balanced for both features.

One of the most important aspects of textual data is the length since different models have different context windows. Figure 1 provides the distribution of text lengths in tokens for the training subset of data. The mean length of text is 471 tokens, minimum – 2, maximum – 34,192. This is es-

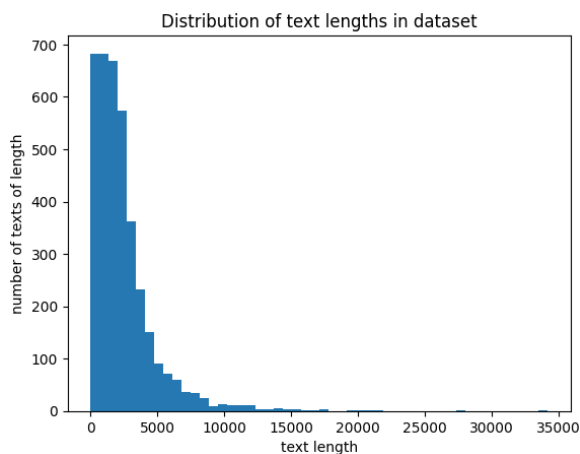


Figure 1: Distribution of text lengths (in tokens)

pecially important for few-shot approaches, since the examples provided to the model need to be short enough to fit in the prompt, while also being around mean to represent the average data. When it comes to fine-tuning, on the other hand, shorter text lengths can negatively impact the models performance, as suggested in the paper introducing the M4 dataset used in our work (Wang et al., 2024b). They have found that for texts shorter than 1,000 tokens F1 scores dropped from 0.99 (for length $\geq 1,000$) to 0.96 (for length = 500). Considering our mean length and the distribution (Figure 1), it is important to consider the effect of this factor on the models' performance as well.

2.2 Related work

In the benchmark (Wang et al., 2024a) two zero-shot approaches were discussed.

The first one was proposed by the team MasonTigers (Puspo et al., 2024). Their approach was based on T5Tokenizer (Raffel et al., 2020) and FlanT5 (Chung et al., 2024) with zero-shot prompting. However, approach proposed by MasonTigers with zero shot architecture was not that successful and their F1 score was equal to 0.49. We think that the result could be improved with an instruction-tuned model. They also did full fine-tuning without quantization / PEFT methods (like LoRA (Devalal and Karthikeyan, 2018) or QLoRA (Dettmers et al., 2024)) on the t5-small (Raffel et al., 2020) model as well and in that case, they achieved a result of F1 equal to 0.57 on the development dataset. During the training, they used an NVIDIA A100 GPU with 80GB.

The other approach with zero-shot which was mentioned in the benchmark paper was proposed by the team KInIT (Spiegel and Macko, 2024). Although they used a zero-shot approach in their methodology, it was used only with statistical methods such as Entropy, Rank, and Binoculars. Their main method used Falcon-7B (Almazrouei et al., 2023) and Mistral-7B (Jiang et al., 2023) fine-tuned models which demonstrated amazing results on multilingual development dataset with F1 score equal to 95.34 and the team took fourth place in the competition. Nevertheless, on the monolingual development dataset their result was not as impressive with F1=79.12 on the development dataset.

2.3 Task formulation

In our case, the task of artificial text detection can be formulated as a binary classification prob-

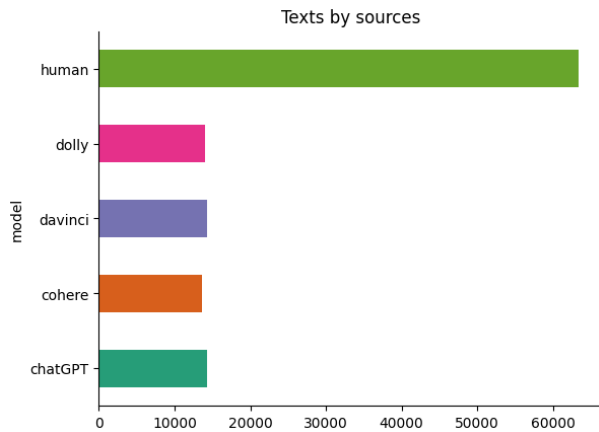


Figure 2: Distribution of models

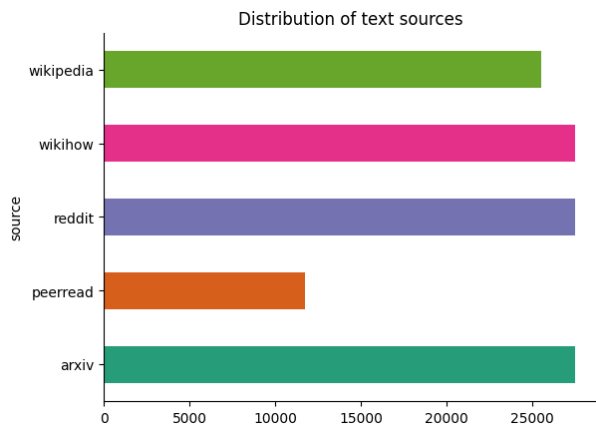


Figure 3: Distribution of sources

lem, where the goal is to determine whether a given text input is artificially generated or human-written. Formally, let X be the input text, and Y be the binary label indicating whether the text is artificial ($Y=1$) or real ($Y=0$).

We frame this task as a textual entailment problem, where the input text X is treated as the premise, and two hypotheses are constructed: H1: "The given text is artificially generated." H2: "The given text is human-written."

3 Fine-tuned model

As the baseline, we fine-tuned the gemma-2b (Team et al., 2024) model on 300 examples from M4 corpus described in 2. The training process was conducted on GPU P100 and took around 40 minutes on 2 epochs. We applied early stopping and 2 epochs were enough for the model to converge.

It is worth mentioning, that amount of training instances play a significant role in the accuracy of predictions. During the fine-tuning stage we have experimented with different sizes of training dataset and we mentioned that the model that was fine-tuned on 100 instances performed with an F1 score equal to 55 but the model that was fine-tuned on 300 instances performed dramatically better with an f1 equal to 0.74. Nevertheless, we fine-tuned our model on only 300 instances in comparison to MaisonTiger t5-small which was trained on the full train corpus, we achieved much better results on development dataset with F1 equal to 0.74. But we have to mention that our model has dramatically more parameters than t5-small and we used LoRA PEFT method which allowed us to fine-tune the model really fast and efficiently.

4 Few-shot with instruct fine-tuned LLMs

Few-shot prompting can be leveraged for artificial text detection tasks using LLMs. The approach has various advantages: mainly the ability to generate prediction with no labeled data, no training time, and no parameter-tuning, all of which would be necessary for fine-tuning. Additionally, it is less resource-intensive and does not require a technical understanding of model architecture. On the other hand, few-shot approaches are extremely sensitive to prompt formulation and usually perform well on more general tasks, e.g. translation (since such examples are likely to be present in the training data). The task of artificial detection is more complicated for the model to grasp without additional training on language- and domain-specific data since it is less likely to appear in the training set for both raw and instruction-tuned data.

We have conducted experiments for both zero-shot and few-shot approaches. For the zero-shot setup, 3 different prompts were tested to ensure diversity.

List of prompts:

1. "You need to determine whether a text is human-written or artificially generated. Answer 0 if the text is human-written and 1 if it is not. DO NOT write anything else, only 0 or 1. Text: "
2. "Is the text written by a human or generated by AI? Write 0 if you think it was written by a human, write 1 if you think it is AI-generated. DO NOT write anything else, only 0 or 1. Text: "
3. "Given an article, classify it into one of two classes: [0,1], where 0 – text is written by a

human, 1 – text is generated artificially. DO NOT write anything else, only 0 or 1. Article:
"

Each prompt was inserted into an instruct-formatted text together with the article text:

```
<s>[INST]{prompt}\n {article}\[/INST]
```

The Mistral-7B-Instruct-v0.2 model (Jiang et al., 2023) was then passed the prompt to generate predictions, with a max of 2 new tokens.

For the few-shot setup, one example for each class was selected randomly. The lengths of both examples were around the mean length in the dataset. The prompt was then passed in the following format:

```
"Given a text, answer 0 if it's human-written or 1 if it is AI-generated.  
DO NOT write anything else, only 0 or 1.  
Example:  
[Text]: ...  
[Label]:0  
[Text]: ...  
[Label]:1  
[Text]: {article}  
[Label]:"
```

5 Results

Table 1 gives an overview of the results based on accuracy, precision, recall, and F1-score. Clearly, fine-tuning has yielded dramatically better results than zero-shot and few-shot approaches. Across different prompts, zero-shot had a mean accuracy of 0.5031 (std=0.0056) and a mean F1-score of 0.3531 (std=0.0131).

The poor performance of few-shot setups is clearly visible from confusion matrices in Figure 4&5. The model is incredibly biased to classify any text as human-written, and only rarely predicts AI-generated label. Interestingly, there was a similar finding in (Wang et al., 2024b) in respect to the BLOOMZ generator specifically: all other generators showed decent results, while BLOOMZ had the worst performance score and simultaneously the lowest recall for label 1 (AI-generated). This may suggest that detectors could be biased to classify text as human-written in nature. Moreover, both for zero-shot and few-shot approaches it is more likely to tag human-written text as AI-generated than predict AI-generated correctly. This

could suggest that the behavior is rather random, with the model tagging articles as AI-generated 13% of the time in a few-shot setup, and 1.5% of the time in a zero-shot setup. This suggests that even an instruction-tuned model does not have an understanding of artificially generated texts, while a fine-tuned model (even on a small subset of samples) shows decent results.

One of the factors which can impact the performance quite significantly is the domain of the data. As has been mentioned earlier, the dataset contains texts from 5 different sources: Wikipedia, Wikihow, Reddit, Arxiv, and peerread. In order to see the effect of domain-specific data, table 2 provides confusion matrices for different approaches across domains. The three rows are approaches (zero-shot, few-shot, and fine-tuned) used, while the 5 columns are data sources / domains (Wikipedia, Wikihow, Reddit, Arxiv, and Peerread).

The best result seen in the table is the performance of the fine-tuned model on Wikipedia texts (third row, first column): the confusion matrix shows strong contrast between correctly predicted instances (top left – true negative, and bottom right – true positive) and mistakes (bottom left – false negatives, top right – false positives). Being the most sensible result, it is a good point of comparison for other matrices. For instance, the entire top row shows the same outline – strong contrast between the left and right half of the matrix, meaning the model almost exclusively outputs 0 as the prediction (i.e. is biased to label texts as human written, as we have discussed above). The few-shot setup results depicted in the second row demonstrate a similar picture except for Arxiv data: the model was more likely to label human-generated texts as AI than correctly, while almost exclusively labeling AI-generated data as human-written. Although the most obvious explanation for such performance is domain specificity of the texts (academic writing style), the same data with a fine-tuned model shows incredibly sensible results. Finally, in the third row – results of the fine-tuned model – the matrices are the most sensible, with exceptions for Reddit and Peerread data, where the model was actually biased to label texts as AI-generated. It is important to note that the samples used for fine-tuning the model were randomly selected from the dataset and did not exhibit a disproportional ratio in domain representation. We thus conclude that there is no domain that would consistently impact the performance for all approaches, and bias towards a

Approach	Accuracy	Precision	Recall	F1
Fine-tuned (300 instances)	0.7424	0.7512	0.7424	0.7401
Zero-shot1	0.4956	0.4307	0.4956	0.3409
Zero-shot2	0.5046	0.6289	0.5046	0.3472
Zero-shot3	0.5092	0.5772	0.5092	0.3713
Few-shot	0.4616	0.4156	0.4616	0.3764
MasonTigers(FLAN-T5 prompting)	0.49	-	-	-
KINIT	0.7271	0.6614	0.9844	0.7912

Table 1: Comparison of Different Approaches

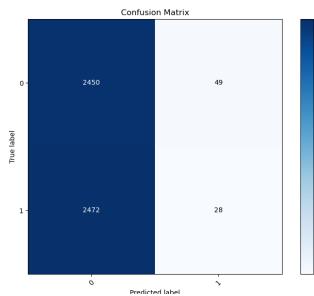


Figure 4: Zero-shot CM

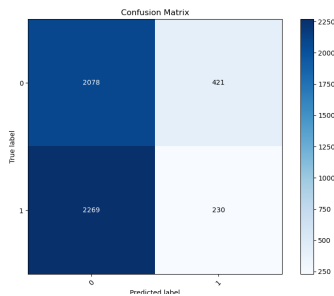


Figure 5: Few-shot CM

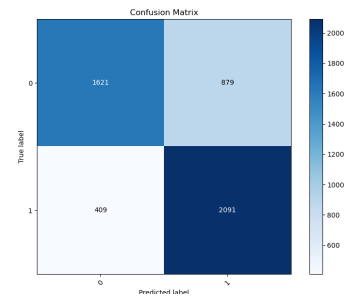


Figure 6: Fine-tuned CM

particular label is also specific to the approach used: zero-shot and few-shot setups are biased to classify texts as human-written, while the fine-tuned model is biased to classify texts as AI-generated for certain domains.

In conclusion, the results show that performance does indeed vary across domains. Moreover, the bias exhibited by the model are dependant on both the approach and the domain, suggesting that both factors need to be kept in mind when interpreting the results of artificial text detectors.

6 Discussion

Although the zero-shot setup can be deemed ineffective for our task, Table 2 suggests that the few-shot setup has potential while being heavily influenced by the provided examples. However, upon inspection of performance across genres, we have found that the difference in performance between zero-shot and few-shot setups is only within data from Arxiv. Moreover, the confusion matrix of performance of the few-shot setup on texts from Arxiv suggests that it is actually less sensible than for other genres. We thereby have to conclude that both zero-shot and few-shot setups have proven completely ineffective in our task, even with instruction-tuned models.

The fine-tuned model shows the best results both overall and upon inspecting domain-specific results. However, there is a significant difference in perfor-

mance across domains for the fine-tuned model: while Wikipedia and Arxiv data demonstrated the best results, other sources of data vary in observed performance. This suggests that the domain of data is incredibly important even for a model that demonstrates satisfactory results on the entire corpus: for example, the model was extremely biased to label Reddit text as AI-generated. The zero-shot and few-shot approaches also demonstrate bias, but in the opposite direction (to label every text as human-generated). We believe that it should be part of a bigger conversation of bias in artificial text detectors.

For instance, a study by researchers at Stanford University revealed that popular GPT detectors tend to falsely label texts written by non-native English speakers as AI-generated. The researchers evaluated seven widely used GPT detectors on 91 essays written by non-native English speakers for the TOEFL (Test of English as a Foreign Language) exam. More than half of these essays were incorrectly classified as AI-generated by the detectors, with one detector flagging nearly 98% of the essays as AI-written. The reason for this bias lies in the way these detectors evaluate text perplexity, which measures how surprising or uncommon the word choices are in a given text. Non-native English writers often use simpler and more common vocabulary, resulting in lower perplexity scores that are more likely to be flagged as AI-generated by the detec-

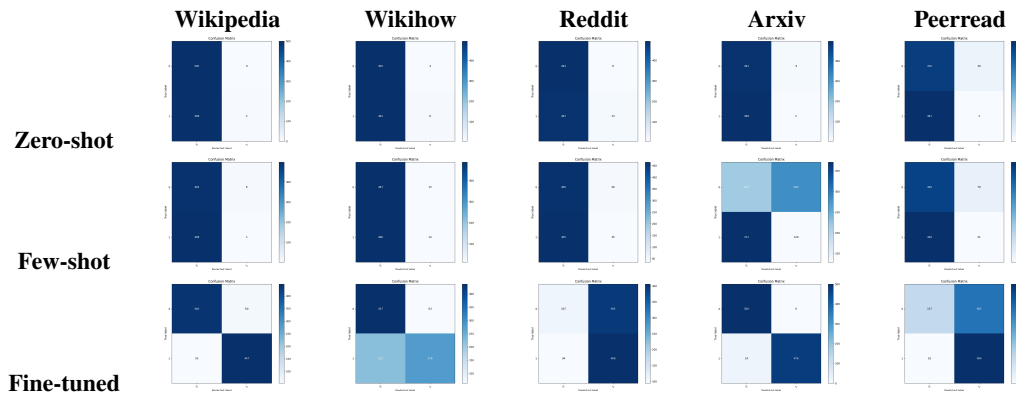


Table 2: Overview of performance across approaches and domains

tors (Liang et al., 2023). The approaches we have explored in this paper are not perplexity-based, and would therefore have other mechanisms responsible for the bias. However, as we have found in this paper, non-perplexity based methods can also be biased in both directions, depending on the domain of data and the approach used.

As has been mentioned in 2.1, the length of texts may also impact performance when it comes to artificial text detection. Considering that the mean length of text in our dataset (471 tokens) is significantly lower than the cutoff found in (Wang et al., 2024b) to negatively impact performance (1000 tokens), we believe it can also be worth investigating, especially in regards to zero-shot and few-shot setups.

Finally, we believe it is worth investigating the results for each generator individually. As depicted in figure 2, the dataset is balanced in regards to the generator used to produce artificial, 4 generators represented: dolly, davinci, cohere, and chatGPT. Considering that (Wang et al., 2024b) found significant differences in performance across generators of their fine-tuned RoBERTa detector, we believe it could be impactful in our case as well.

7 Conclusion

In this paper we have attempted to leverage instruction-tuned LLMs for few-shot detection of artificially generated texts. We have found that both few-shot and zero-shot approaches are unable to surpass the baseline, and are incredibly biased to classify texts as human-written. Nonetheless, the fine-tuned model, while being optimized with LoRA and trained on only a small subset of instances, significantly surpassed the performance of both zero-shot and few-shot setups. We have also

found that the domain of data significantly affects the model performance, especially in the case of the fine-tuned model, which was biased to classify texts from Reddit and Peerread as AI-generated. We believe other factors such as the generator used and the length of texts could be investigated to further explore the bias of AI-detectors.

References

- Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, M erouane Debbah,  tienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. *The falcon series of open language models*. *Preprint*, arXiv:2311.16867.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.
- Shilpa Devalal and A Karthikeyan. 2018. Lora technology-an overview. In *2018 second international conference on electronics, communication and aerospace technology (ICECA)*, pages 284–290. IEEE.
- Europol. 2022. Facing reality? law enforcement and the challenge of deepfakes, an observatory report from the europol innovation lab, publications office of the european union, luxembourg. Technical report, Publications Office of the European Union.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L elio Renard Lavaud,

- Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. [Mistral 7b](#). *Preprint*, arXiv:2310.06825.
- Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou. 2023. [Gpt detectors are biased against non-native english writers](#). *Preprint*, arXiv:2304.02819.
- Sadiya Sayara Chowdhury Puspo, Md Nishat Raihan, Dhiman Goswami, Al Nahian Bin Emran, Amrita Ganguly, and Ozlem Uzuner. 2024. Masontigers at semeval-2024 task 8: Performance analysis of transformer-based models on machine-generated text detection. *arXiv preprint arXiv:2403.14989*.
- 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.
- Michal Spiegel and Dominik Macko. 2024. Kinit at semeval-2024 task 8: Fine-tuned llms for multilingual machine-generated text detection. *arXiv preprint arXiv:2402.13671*.
- Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.
- Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, et al. 2024a. Semeval-2024 task 8: Multidomain, multimodel and multilingual machine-generated text detection. *arXiv preprint arXiv:2404.14183*.
- Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Toru Sasaki, Thomas Arnold, Alham Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024b. [M4: Multi-generator, multi-domain, and multilingual black-box machine-generated text detection](#). In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1369–1407, St. Julian’s, Malta. Association for Computational Linguistics.
- Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

5 Predicting a Prompt from a Generated Text track

Exploring Prompt Prediction: Methods, Performance, and Challenges

Dennis Adelled

University of Oslo / Blindernveien 11

dennis.adelved@geo.uio.no

Abstract

Generative AI advancements have enabled the creation of high-quality texts using pre-trained language models. However, the task of prompt prediction, retrieving the original prompt used to generate a specific piece of text remains under-explored. In this paper, we investigate the prompt prediction tasks from a sequence-to-sequence and a sequence classification approach. We experiment with generating synthetic data, using a pre-trained language model to generate synthetic prompts-response pairs in a zero-shot configuration and demonstrate that the addition of the synthetics can improve the generalization to other, unseen instruction datasets. Our findings suggest that predicting the exact prompt used to generate a text can be challenging, especially for longer prompts. However, we are able to extract the overall intent/sentiment of the prompts.

1 Introduction

The rapid advancements of generative AI have significantly altered the landscape of artificial intelligence, allowing for the generation of high-quality texts. This progress can mainly be attributed to the development and utilization of pre-trained Language Models (LMs). These LMs are initially trained on vast amounts of textual data and can learn robust and general features of the languages they model on consumer-grade hardware. Some of these pre-trained LMs are released as open-source, which can be accessed, fine-tuned and shared using the Hugging Face transformers library and community platform (Wolf et al., 2020). Other companies develop proprietary pre-trained models, such as OpenAI’s ChatGPT (OpenAI, 2021) and release them as a service. This is often referred to as Language Models as a Service (LMaaS), which allows users to interact with the pre-trained model through some specific API to solve various language tasks by specifying task-specific prompts (Sun et al.,

2022). Utilizing in-context learning, these models can be fine-tuned with specific training examples provided by the user in few-shot or zero-shot configurations (Brown et al., 2020). This has made the use and training of language models accessible to the general public, allowing individuals and actors to utilize LMs to generate content for various use purposes. However, this also gives rise to malicious uses such as the generation of fake news (Zellers et al., 2019) or fake product reviews (Adelani et al., 2019). This has given rise to the field of artificial text detection, which aims to mitigate and combat the effectiveness of malicious use of generative AI, by identifying if a text have been written by a human or been generated by a machine (Crothers et al., 2023). The prompt prediction task, which involves predicting the prompt used to generate a specific piece of text, is an under-explored area in natural language processing. This task reverses the usual workflow of predicting a response from a prompt. By understanding the relationship between a text and its originating prompt, we can develop a system to extract the best prompts for generating a reference text. For instance, one could identify the best prompts for generating viral tweets by extracting the prompts from a set of viral reference tweets. Another use case could be uncovering the original intent behind a text, which would be useful in detecting and preventing malicious content.

In this work, we aim to investigate **prompt prediction** using two different approaches. First, we tackle the problem as a sequence-to-sequence task, fine-tuning two different pre-trained model architectures (encoder-decoder and decoder-only) on prompt-response pairs. Secondly, we approach the problem as a sequence classification task, where we aim to classify the prompt category from the text. We compare our trained model using a instruction-tuned pre-trained Llama3-8B-Instruct (AI@Meta, 2024) in a zero-shot configuration. Given the scarcity of open-source instruction datasets, we

generating synthetic prompt-response pairs by paraphrasing the original prompts and respond to them using a pre-trained model in a zero-shot configuration. We have achieved promising results in the sequence-to-sequence approach, with our best model reaching a BERT score of 0.925 and a Rouge-L score of 0.511 on the test set. We also show that the inclusion of synthetic prompt-response pairs can be beneficial when generalizing to other datasets. Furthermore, we observe that exact prompt prediction is both challenging and biased towards shorter prompts. However, we can demonstrate that the models can capture the general intent of the prompt that was used to generate a text.

2 Related work

To the author’s knowledge, there are currently no publications on predicting the prompt that was used to generate a text. However, several related areas of research can provide useful insight and serve as inspiration. One such area is prompt engineering, which often requires extensive manual work to obtain reasonable results in zero-shot and few-shot configurations (Liu et al., 2021). To reduce this burden, research has been working towards automating this process through automatic prompt-tuning. The goal is to optimize the prompts, such that when combined with the pre-trained LM, yields the highest performance on a given task. This has been illustrated using both gradient-based prompt-tuning of parameterized prompts (Lester et al., 2021), or by training a light-weight prompt generation model to enhance the in-context learning if a target LM (Ha et al., 2023). Another similar field of research is artificial text detection, where the goal is to identify machine-generated texts with the use of AI. This problem is often primarily formulated as a supervised binary classification task, where input texts are either classified as human or machine-generated (Wang et al., 2024). Although research on directly predicting prompts from a generated text has been limited, there seems to be a growing interest in the topic. For instance, earlier this year, Kaggle posted a competition for recovering the prompt that was used to stylistically rewrite a given text. E.g., "Rewrite this paragraph but do it using the writing style of J.R.R. Tolkien" (Will Lifferth, 2024).

3 Methodology

3.1 Dataset

We train and evaluate our models using the databricks-dolly-15k dataset, which is an open-source dataset of instruction-following records, consisting of 15,000 human-made prompts in eight different instruction categories in English (Conover et al., 2023). The categories in the dataset includes: closedQA, classification, openQA, information extraction, brainstorming, generalQA, summarization, and creative writing. The dataset is unbalanced, with an over-representation of the openQA. Thus, we perform a stratified split in the train-development and test set, ensuring a similar representation of each category in both sets. Due to data scarcity, we limit the test size of 0.1 to maintain as much data as possible for training and development, resulting in a train-development/test set of 13,500/1500 samples. The test set will be locked away and not touched until the final model evaluation. General dataset statistics and the prompt category distribution can be found in Appendix A.

3.2 Synthetic Data

We generate the synthetic training data in the following way: first, we use the pre-trained Llama3-8B-Instruct in a zero-shot configuration to paraphrase the instruction prompts in the train-development set. The paraphrased prompts are then fed back into the model to generate synthetic responses to the paraphrased synthetic prompts, which will effectively double the number of prompt-response pairs at our disposal. The synthetic generation process can be seen in Figure 1. The details

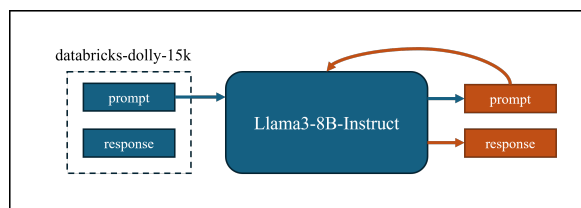


Figure 1: Synthetic zero-shot generation pipeline. The original dataset is displayed in blue and the synthetic prompt-response pairs displayed in orange.

concerning the exact generation of the paraphrased prompts, and the generation of the model responses is further outlined in Section 4.1.

3.3 Sequence-to-Sequence Approach

We consider two architectures (encoder-decoder and decoder-only) for prompt prediction to evalu-

ate if there is any difference based on architecture type. Since we aim to instruct our model to retrieve the prompt used to generate a specific piece of text, we opt for pre-trained language models that have been further fine-tuned on a wide range of instructions, as they provide a better starting point for learning the prompt prediction task. For the decoder-only architecture, we use the Llama3-8B-Instruct, which is a pre-trained Llama3 with 8B parameters, which have been instruction tuned (AI@Meta, 2024). To be comparable to the 8B Llama3, we must select a sufficiently large encoder-decoder model, such as the FLAN-T5 XL model with 2.85B parameters, which is pre-trained and instruction-tuned on a wide variety of tasks (Chung et al., 2024). We compare the fine-tuned models to a Llama3-8B-Instruct in a zero-shot configuration, which we name the Baseline model. The models in the sequence-to-sequence approach is summarized in Table 1.

Given the relatively recent release of these models, we must consider the possibility of data leakage, where the data used in our experiment might have been included in the pre-training of the models. The knowledge cutoff for the Llama3-8B-Instruct model is March 2023¹, and the most recent dataset used in the instruction fine-tuning of the FLAN-T5-XL model is from 2022 (Chung et al., 2024). The official release of the Databrick-Dolly-15k dataset was in April 2023², making it reasonable to assume that this dataset was not included in the pre-training of either model.

Name	Configuration
FLAN-T5-XL	fine-tuned
Llama3	fine-tuned
Baseline (Llama3)	zero-shot

Table 1: The models used in the sequence-to-sequence approach. Llama3 refers to the Llama3-8B-Instruct.

Evaluation To evaluate the model’s ability to predict prompts, we employ two primary approaches. First, we assess the model’s capability to exactly reconstruct the prompt, utilizing the Rouge-L score as a metric. The Rouge-L score measures the longest common sub-sequence between the prediction and the reference (Lin, 2004). Secondly,

¹<https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct>

²<https://github.com/databricks/dolly/commit/d000e3030970379aabbf6d291f50ffdd3b715b64>

we evaluate the model’s ability to capture the general intent of the prompt. For this task, we use the BERT score, which compares the similarity between the prediction and reference embeddings, providing an evaluation of the semantic similarity between the two texts (Zhang* et al., 2020).

Quantization and Low-Rank Adaptation

Since we are limited to a single GPU setup, we find it necessary to quantize both the 2.85B FLAN-T5-XL and the Llama3-8B-Instruct from their respective float32 and float16 into 4-bit. The quantized 4-bit FLAN-T5 XL can be trained in its entirety on a single GPU. However, the 4-bit Llama3-8B-Instruct must be fine-tuned using Low-Rank Adapter weights (LoRA) (Hu et al., 2022). To achieve comparable results to the float 16 models in a 4-bit setting, we follow the Quantized Low-Rank Adapter (QLoRA) approach outlined by Dettmers et al. (2023), using the same LoRA configurations for their small LLMs case (7-13B parameters) and adding LoRA modules to all linear layers of the base models with LoRA parameters: $r=16$, $\alpha=64$ and $\text{dropout}=0.01$.

3.4 Sequence Classification Approach

We aim to investigate the prompt prediction task from a classification perspective. As detailed in Section 3.1, the databrick-dolly-15k are labeled into 8 distinct prompt categories: closedQA, classification, openQA, information extraction, brainstorming, generalQA, summarization, and creative writing, which will serve as our classes. By using an encoder-only transformer, we can process input sequences and generate a fixed-size representation that matches the number of classes in our dataset. One such model is the BERT model, which is based on a multi-layer bidirectional transformer that has been trained on masked word prediction tasks (Devlin et al., 2019). We use the bert-uncased-base pre-trained model (110M parameters), with 12 transformer blocks, 12 attention heads, and a hidden size of 768, which has been pre-trained on lower-cased English text (Turc et al., 2019) as the base model, and attach a classification head that converts the encoder’s output into the desired number of classes. The model will be evaluated using the macro F1-score, which is the harmonic mean of precision and recall. This metric is particularly useful for imbalanced datasets as it provides a balanced measure of performance across all classes.

4 Experiment

4.1 Synthetic Data Generation

Both in the paraphrasing and in the synthetic response generation we use the default sampling parameters with $temperature = 0.6$ and $top_p = 0.9$ (AI@Meta, 2024). To get desirable output in a zero-shot configuration, we follow the Llama3-8B-Instruct chat template, where we provide some initial instruction through the system role, before prompting the model to paraphrase the user’s input or to generate a synthetic response. The prompts, with the initial system instruction and the user input instruction is detailed in Appendix B. Due to the generation process being highly time-consuming, we were only able to generate synthetic data for 11.500 samples of the 13.500 samples, resulting in a total train-development dataset of 23.0000 samples.

4.2 Finetuning Sequence-to-Sequence

We conducted our fine-tuning experiment for the sequence-to-sequence using the combined prompt-response (synthetic + real) pairs. Based on the tokenized dataset statistics (Appendix A), we set the maximum prompt/response lengths to 128/384 for both the Llama3-8B-Instruct and FLAN-T5-XL model. The train-development split was set to 0.1. We use the same prompt template for both models and is as follows:

```
"Predict the prompt that was used to  
generate the following text:{response}"
```

Both models are loaded and fine-tuned using the same quantization outlined by Dettmers et al. (2023), using 4-bit NormalFloat, with double quantization and bf16 computation datatype.

Llama3-8B-Instruct We add LoRA modules on all linear layers with the LoRA parameters $r=16$, $\alpha=64$ and $dropout=0.01$. We use an AdamW optimizer (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a weight decay of 0.01. The learning rate is set to 5×10^{-5} and we use a linear learning rate schedule with 100 warmup steps. We used a batch size of 16 and trained the model for 2 epochs which took around 12 hours on an A100 GPU, saving the best performing model on the development set.

FLAN-T5-XL The FLAN-T5-XL is trained in its entirety, without LoRA. We use an AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a weight

decay of 0.01. We experiment with four learning rates: 1×10^{-5} , 2×10^{-5} , 3×10^{-5} , and 5×10^{-5} . In all experiments, we use a linear learning rate schedule with 100 warmup steps. The batch size is set to 2 and the model for 4 epochs which took around 3 hours on an A100 GPU, saving the best-performing model on the development set. To facilitate the use of sequence-to-sequence generation after fine-tuning, we use the **T5ForConditionalGeneration**³ class, which attaches a language modelling head on top of the TF5 architecture. Using the same setup as outlined above, we also fine-tune three models with learning rate 1×10^{-5} , 3×10^{-5} , and 5×10^{-5} on the real dataset only, to observe the effect of generating synthetic prompt-response pairs.

4.3 Finetuning Sequence Classification

The pre-trained BERT and the classification head are fine-tuned using the same dataset as before, consisting of 23.000 samples using a 0.2 development split. Based on the tokenized dataset statistics (Appendix A), we set the max sequence length to 128, which includes most of the prompts in the dataset. We use an AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a weight decay of 0.01. To avoid catastrophic forgetting we initialize the training with learning rate 1×10^{-5} , similar to the one found effective by Sun et al. (2020) and we use a linear learning rate schedule with 2000 warmup steps. The batch size is set to 24, and we train four models with dropout rates of 0.1, 0.2, 0.3, and 0.5. Dropout is applied to the embedding layer, the self-attention mechanism within each encoder layer, the attention output, and the feed-forward network output. Additionally, dropout is applied to the classification head. This regularization strategy aims to improve generalization and prevent overfitting.

5 Results

5.1 Quality of Synthetic Zero-Shot Prompts

Ideally, a good paraphrased prompt should preserve the meaning of the original prompt, while being different enough to introduce diversity to the dataset. This corresponds to a high BERT score with a low to moderate Rouge-L score. To get an idea of the generated prompt quality, we analyze the Rouge-L and BERT scores between the original and paraphrased prompts for each category. The

³https://huggingface.co/transformers/v2.10.0/model_doc/t5.html#t5forconditionalgeneration

BERT score for each category is high (greater than 0.9), while the Rouge-L score is moderate ($0.5 > \text{Rouge-L} > 0.3$) (Appendix C.1), aligning well with our notion of a good synthetic prompt. While both metrics are important, the BERT score seems to better reflect a paraphrased prompt’s usefulness, as the intent can be preserved even without any common sub-sequence.

Low Rouge-L values appear to be detrimental to prompt quality only if the BERT score is also low. This combination is associated with the lowest prompt quality, where the model tends to respond to the input prompt instead of paraphrasing or omits essential information, such as the list of objects to classify in a classification prompt. Moderate BERT scores are associated with generally useful prompts, especially when combined with moderate Rouge-L scores. These prompts typically exhibit increased verbosity, often in the form of over-specification. High BERT scores paired with a low to moderate Rouge-L score indicates the highest quality prompts, since they introduce the most diversity to the dataset while preserving the meaning of the original input. Having both a high BERT score and Rouge-L score also produces good prompts. However, they are often very similar to the original prompt, where often only single words are changed. Finally, when both metrics approach 1, the result is closer to generating duplicates, which is undesirable as these synthetic prompts introduce redundancy rather than diversity. Assigning numerical cutoffs for synthetic prompt quality is challenging and varies by prompt category. Therefore, we include all prompts in the training without setting specific cutoffs. However, representative values and examples for the different types of prompt quality discussed can be found in Appendix C.2.

5.2 Sequence-to-Sequence

The training-development loss for the FLAN-T5-XL experiments is detailed in Appendix D.1. In both scenarios, a lower learning rate of 1×10^{-5} yields the best model, seemingly converging without overfitting. To distinguish between the two, we will refer to the encoder-decoder models as **FLAN-T5-XL-S** and **FLAN-T5-XL-R** for the FLAN-T5-XL trained on the combined (synthetic + real) dataset and the real dataset, respectively.

The training-development loss for the Llama3-8B-Instruct model is detailed in Appendix D.2. Due to the extended training duration, we could

not evaluate both models over the same number of epochs, which is a limitation. No overfitting behaviour was observed, suggesting that the model could benefit from further training.

Generation Using the prompt template from Section 4.2, we generate predictions for the development and test sets with the same sampling parameters used for synthetic data in Section 4.1. Additionally, we generate predictions on the test set using greedy decoding to compare the two methods and to provide a deterministic evaluation metric.

Evaluation The evaluation of our results is divided into two distinct sections. First, we will assess the models trained on the combined dataset, specifically FLAN-T5-XL-S and Llama3-8B-Instruct, and compare them to the zero-shot Baseline model using both the development set and the test set. Secondly, we will evaluate the impact of the synthetic data by comparing FLAN-T5-XL-S to FLAN-T5-XL-R on the test set and on 8000 random samples from the Alpaca dataset⁴, which is a instruction dataset generated using OpenAI’s *text-davinci-003* engine (Taori et al., 2023).

The evaluation results for the best-performing FLAN-T5-XL-S, the Llama3-8B-Instruct, and the Baseline model on the development and test set are presented in Table 2. The FLAN-T5-XL-S achieves the highest BERT and Rouge-L scores, followed by the fine-tuned Llama3-8B-Instruct, with the zero-shot Baseline model. All models exhibit high BERT scores on both the development and test sets, with only a small relative difference between the two. In contrast, the Rouge-L scores show a significantly higher relative difference between the development and test sets.

The comparison in Table 3, shows that FLAN-T5-XL-R outperforms FLAN-T5-XL-S on the test set, while FLAN-T5-XL-S outperforms FLAN-T5-XL-R on the Alpaca test set.

5.3 Sequence Classification

The training and development losses for the four models trained with the hyperparameters detailed in Section 4.3 which can be found in Appendix D.3. To address the overfitting observed with the default dropout rate of 0.1, we conducted experiments with varying dropout rates. We use model checkpointing, only saving the best-performing model on the

⁴Released March 2023. Leakage should not be a problem with FLAN-T5-XL https://github.com/tatsu-lab/stanford_alpaca/commits/main/alpaca_data.json

Model	BERT score	Rouge-L
<i>Development set</i>		
Baseline	0.843	0.155
Llama3-8B-Instruct	0.907	0.463
FLAN-T5-XL-S	0.942	0.618
<i>Test set</i>		
Baseline	0.835	0.125
Llama3-8B-Instruct	0.897/0.892	0.387/0.362
FLAN-T5-XL-S	0.923/0.921	0.492/0.483

Table 2: BERT score and Rouge-L metrics for the two fine-tuned models and the Baseline model are provided. Entries separated by a slash, shows scores derived from texts generated using sampling on the right-hand side and greedy decoding on the left-hand side.

Model	BERT score	Rouge-L
test set		
FLAN-T5-XL-S	0.923	0.492
FLAN-T5-XL-R	0.925	0.511
Alpaca		
FLAN-T5-XL-S	0.908	0.441
FLAN-T5-XL-R	0.906	0.422

Table 3: BERT score and Rouge-L metrics for FLAN-T5-XL-S and FLAN-T5-XL-R using greedy decoding.

development set. Although increasing the dropout rate across most layers helped reduce overfitting, the lowest development loss was still achieved with a dropout rate of 0.1 at epoch 3. This suggests that applying dropout uniformly across all layers may be too aggressive that negatively impact model performance. Following our model selection strategy we select the model with dropout rate of 0.1, with the checkpoints from epoch 3. The calculated macro F1-scores on the development and test sets are **0.641** and **0.639**, respectively.

6 Discussion

In this section, we will take a closer look at the results discussed in Section 5, analyzing the differences observed in the sequence-to-sequence approach w.r.t. model architecture and inclusion of synthetic data. Finally, we examining the poor performance noted in the classification task.

6.1 Key Findings: Sequence-to-Sequence

The FLAN-T5-XL models achieves the highest evaluation metrics across both BERT score and Rouge-L among the models in the sequence-to-sequence approach. The fine-tuned Llama3-8B-

Instruct model also performs well, especially when compared to the zero-shot Baseline. Additionally, the generation method shows minimal impact on evaluation metrics, with deterministic greedy decoding having a slight edge over the default generation parameterization.

Model Architecture and Limitations Since FLAN-T5-XL is smaller than the Llama3-8B-Instruct model, we can train the full model (in 4-bit quantization) without using parameter-efficient tuning methods like LoRA, which complicate model comparison. A significant limitation is the longer training time for Llama3-8B-Instruct, with each epoch taking approximately 3-4 times longer than FLAN-T5-XL, even with QLoRA (4-bit quantization). Therefore, we are cautious about making strong claims in comparing the two models, although their test set performances are relatively similar and both outperform the Baseline model. One main advantage of FLAN-T5-XL is its smaller size, allowing for greater flexibility and more frequent experimentation. Additionally, FLAN-T5-XL benefits from being an encoder-decoder model, which is conceptually simpler and more intuitive to train compared to a decoder-only model.

Real vs Synthetic Comparing the fine-tuning of FLAN-T5-XL-S and FLAN-T5-XL-R models, with and without synthetic data, reveals that including synthetic data slightly degrades performance on the test set for both metrics. The difference in BERT score is marginal, and illustrates that both models are able to capture the intent of the original prompt. The impact observed on the Rouge-L score is much larger, which shows that the inclusion of synthetic samples degrades the models ability to capture the structure of the original prompts. As shown in Figure 2 the Rouge-L score differences are most pronounced in the information extraction, summarization, and QA categories. These categories typically feature short, concise prompts that are difficult to paraphrase. This often results in overly verbose and complex formulations, sometimes including partial answers, which are significantly different in structure to the original prompts. Occasionally, the model ignores the paraphrasing and directly responds to the input prompt, which seems to be more frequent in these categories. Examples of this behavior are included in the Appendix E.

Apart from the direct responses, we believe that both the partial responses and verbose/complex

paraphrasings are valuable for learning, as they increase the diversity of the dataset. This is supported by the evaluation on the Alpaca dataset, where the FLAN-T5-XL-S model outperforms the FLAN-T5-XL-R model, especially in the Rouge-L score. This suggests that the inclusion of the synthetic data allows for better generalization to datasets with different prompt structures. This further supports the idea that the performance drop observed for the FLAN-T5-XL-S on the test set is likely due to the structural differences in the synthetic prompts compared to the original prompts, rather than the synthetic prompts being of poor quality.

Seeing that the performance on the test set is sensitive to the structure of the synthetic prompts, we suggest that further work should experiment with adding synthetic prompts generated using a few-shot configuration. By providing the model with prompt examples, we may be able produce synthetic prompts that are more structurally similar to those in the original prompts in each category.

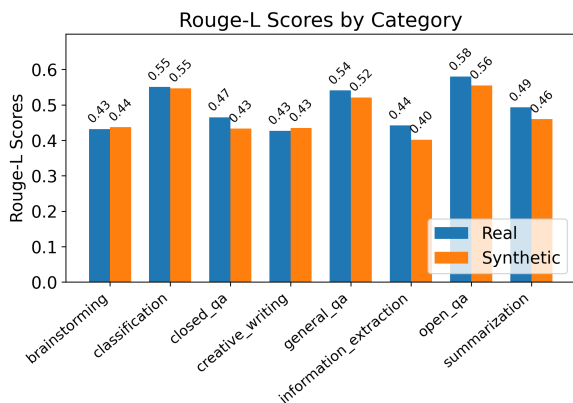


Figure 2: The Rouge-L scores in each prompt category for the FLAN-T5-XL-S and FLAN-T5-XL-R on the test set.

Prediction Quality The definition of a quality prediction varies based on the goal. Sometimes, a perfect match is required, while other times, understanding the prompt’s intent is more important. Since FLAN-T5-XL-R performed best on the test set, we select this model for evaluating the prediction quality. We establish the following criteria and query only the matching predictions:

Strict An exact match where both the Rouge-L and BERT scores are 1, requiring an exact character and case match.

Moderate Rouge-L score must be greater than 0.8. Requires a high degree of match in common

sequence.

Lenient BERT score must be greater than 0.8. No requirement for a common sequence, only semantic similarity.

Using the predicted prompts with greedy decoding, around 10% of the predictions meet the strictest criterion, with an average token length of 9. This indicates that case sensitivity and exact character match requirements cause the model to struggle, often rejecting valid predictions due to minor differences like missing question marks or incorrect capitalization.

The moderate criterion captures 20% of the predictions, with an average token length of 12. By emphasizing a high Rouge-L score, we retrieve nearly identical prompts, allowing minor deviations in character order.

The lenient criterion, focused on a high BERT score, includes 60% of the predictions with an average token length of 13. This approach prioritizes semantic similarity, making it useful for understanding the general intent of the prompt. From this we gather that, exact prompt prediction is a challenging task, with only 20% meeting our moderate criterion, indicating that our model is not able to exactly retrieve the prompt from the response. From the average token length, it seems like exactly matching prompts are biased towards shorter prompts. This can be related to there being less room for variability in a short piece of text. However, the model seems to be better at matching the general intent of the prompts, where we can retrieve 60% of the predictions of relatively good quality (Appendix F).

Investigating the Low F1-score As noted in Section 5, both the development and test F1-scores for the prompt classifier are lower than expected. Figure 3 shows the confusion matrix for the test set. We identify some categories such as classification and brainstorming to be doing well, with many correct classifications and few misclassifications. The openQA category does have a large amount of correct classifications. However, we observe a significant amount of confusion between openQA and the other QA-categories. Examining the prompts in the QA-categories, we see that some examples are impossible to distinguish, such as the generalQA *"what is a dog?"*, the openQA *"what is a debit card?"* and the closedQA *"what is a Druid?"*. This confusion is strongest between generalQA and

openQA. Similar issues can be observed for summarization and information extraction and the QA-categories.

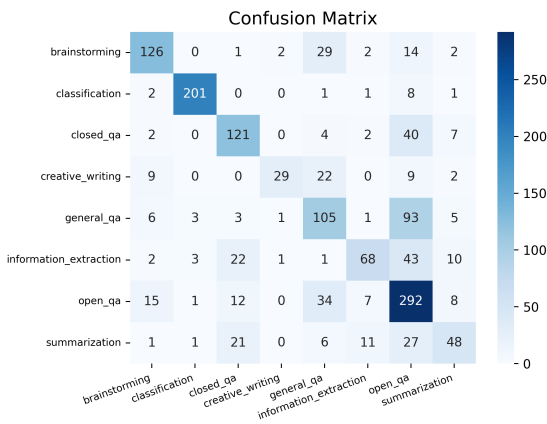


Figure 3: Confusion matrix for the test set. Ground truth/Prediction on the vertical/horizontal axis.

Summarization, information extraction, and closedQA tasks use a supplementary reference text as context⁵. This context is sometimes explicitly mentioned in the prompts, e.g., *"From the passage below."*, which allows them to be separated from openQA and generalQA. However, this also makes these tasks more susceptible to being confused with one another. Finally, the worst-performing category is creative writing. While some prompts in this class are very distinct, others could easily be mistaken for prompts from the QA categories. The low F1-score can be attributed to the fact that the majority of the samples in the dataset belong to classes that are often confused with one another, due to ambiguity in these class definitions.

Suggestion for Improvements One of the main issues with the classification is the significant overlap between some of the prompt categories, making them practically indistinguishable based on the prompt alone. Therefore, any further attempts at prompt classification should focus on curating the dataset to remove ambiguities between the categories. Suggestions to address these issues include: 1) merging openQA and generalQA into a single category, and 2) removing QA-style prompts without reference to the provided context from summarization, closedQA, and information extraction. Since openQA and generalQA are the most frequent categories, this merge will amplify the dataset imbalance. Thus, generating synthetic data

⁵<https://huggingface.co/datasets/databricks/databricks-dolly-15k>

to target underrepresented categories, such as creative writing, can also be considered.

7 Conclusion

In this study, we explored the prompt prediction task using both sequence-to-sequence and sequence classification approach. In the sequence-to-sequence approach we fine-tuned an encoder-decoder model (FLAN-T5-XL) and a decoder-only model (Llama-8B-Instruct). Additionally, we investigated the impact of incorporating synthetic data by generating synthetic prompt-response pairs, paraphrased from our dataset using Llama3-8B-Instruct in a zero-shot configuration. Our experiments revealed that fine-tuning the encoder-decoder architecture produced better and faster results compared to the larger decoder-only model, given our single GPU hardware constraint. The inclusion of synthetic data led to a marginal decline in the BERT score and a more significant decline in the Rouge-L score. We attribute these declines to structural differences between the synthetic prompts and the original dataset, resulting in lower performance on the real test set. However, the synthetic dataset introduced useful data diversity, allowing the FLAN-T5-XL-S model to generalize better on a different instruction dataset not included in the fine-tuning process.

Both the high BERT score (> 0.9) and the marginal difference between the real and synthetic case suggest that extracting the general intent of a prompt can be effectively learned and is less dependent on the dataset compared to learning the structure of the prompts. Our qualitative analysis revealed that approximately 10% of the predicted prompts in the real test set can be regarded as exact matches (under our moderate criterion), highlighting the challenge of predicting exact matches, particularly for longer prompts. The sequence classification approach was less successful, which we largely attribute to the ambiguity in the defined classes when considering only the prompt and the class label.

Overall, our findings suggest potential for further research in prompt prediction techniques, particularly in improving the prediction of exact matches and addressing class ambiguity in classification tasks. We hope this work can serve as inspiration for future studies to refine these methods and overcome the challenges identified in this research.

References

- David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. 2019. [Generating sentiment-preserving fake online reviews using neural language models and their human- and machine-based detection](#). *Preprint*, arXiv:1907.09177.
- AI@Meta. 2024. [Llama 3 model card](#).
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2024. [Scaling instruction-finetuned language models](#). *Journal of Machine Learning Research*, 25(70):1–53.
- Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. [Free dolly: Introducing the world’s first truly open instruction-tuned llm](#).
- Evan Crothers, Nathalie Japkowicz, and Herna L. Viktor. 2023. Machine-generated text: A comprehensive survey of threat models and detection methods. *IEEE access*, 11:1–1.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. [Qlora: Efficient finetuning of quantized llms](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc.
- 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.
- Hyeonmin Ha, Jihye Lee, Wookje Han, and Byung-Gon Chun. 2023. [Meta-learning of prompt generation for lightweight prompt engineering on language-model-as-a-service](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2433–2445, Singapore. Association for Computational Linguistics.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [Lora: Low-rank adaptation of large language models](#). In *ICLR 2022*.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The power of scale for parameter-efficient prompt tuning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. 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.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. [Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing](#). *CoRR*, abs/2107.13586.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *arXiv.org*.
- OpenAI. 2021. Chatgpt: A large-scale generative model for open-domain chat. <https://github.com/openai/gpt-3>.
- Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. [How to fine-tune bert for text classification?](#) *Preprint*, arXiv:1905.05583.
- Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-box tuning for language-model-as-a-service. In *Proceedings of ICML*.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.
- 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:1908.08962v2*.
- Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Chenxi Whitehouse, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024. Semeval-2024 task 8: Multidomain, multimodel and multilingual machine-generated text detection. *arXiv.org*.

Sohier Dane Ashley Chow Will Lifferth, Paul Mooney. 2024. [Llm prompt recovery](#).

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.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. [Defending against neural fake news](#). 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 9054–9065. Curran Associates, Inc.

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*.

A Appendix: Dataset statistics

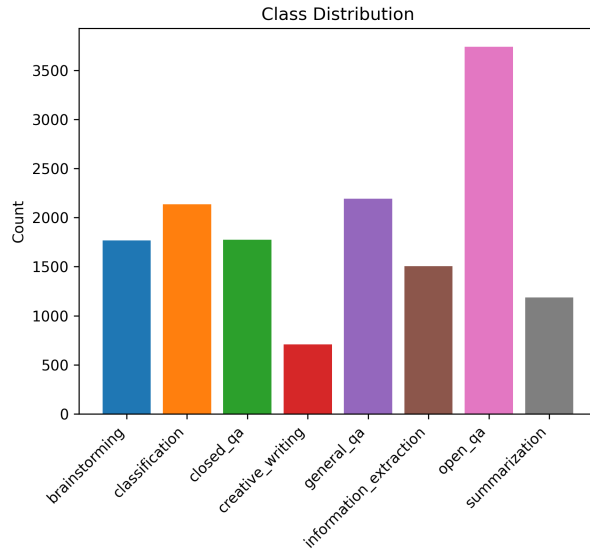


Figure 4: Class distribution of the original databrick-dolly-15k dataset.

Model	Type	Min Length	Max Length	Avg Length	95th Perc	99th Perc
flan-t5-xl	Prompt	1	5924	18	42	78
	Response	1	5636	83	270	589
meta-llama-3-8b-instruct	Prompt	1	5373	16	37	68
	Response	1	5262	77	251	550
bert-base-uncased	Prompt	1	6405	16	37	69
	Response	1	5191	75	247	533

Table 4: Dataset statistics for the real prompts and responses for the different model tokenizers.

Model	Type	Min Length	Max Length	Avg Length	95th Perc	99th Perc
flan-t5-xl	Prompt	1	313	19	38	63
	Response	2	340	93	192	266
meta-llama-3-8b-instruct	Prompt	1	236	17	33	55
	Response	2	256	84	172	256
bert-base-uncased	Prompt	1	261	17	34	55
	Response	2	296	85	175	243

Table 5: Dataset statistics for the synthetic prompts and responses for the different model tokenizers.

Model	Type	Min Length	Max Length	Avg Length	95th Perc	99th Perc
flan-t5-xl	Prompt	1	5924	18	40	71
	Response	1	5636	88	215	428
meta-llama-3-8b-instruct	Prompt	1	5373	16	35	62
	Response	1	5262	80	197	394
bert-base-uncased	Prompt	1	6405	16	35	62
	Response	1	5191	80	197	390

Table 6: Dataset statistics for the combined (synthetic + real) prompts and responses for the different model tokenizers.

B Appendix: Zero-shot Prompt for Synthetic Data Generation

Prompt-paraphrasing	
Role	Content
System	You are an AI system that helps the user to rephrase their prompt. The response must preserve the exact meaning of the user’s prompt. The response should be brief and clear. The response should only include the rephrased prompt, without starting with 'Here are'.
User	Rephrase the following: {prompt}
Generating responses	
System	Provide a single paragraph response without additional details.
Context	{context} If available
User	{synthetic_prompt}

Table 7: Zero-shot prompt structure for LLama3-8B-Instruct. Generating synthetic prompts by paraphrasing the human-made instructions from the databricks-dolly-15k dataset and for generating synthetic responses with the synthetic prompts.

C Appendix: Synthetic Prompt Quality

C.1 BERT score and Rouge-L score across the different prompt categories

Category	BERT score	Rouge-L
Brainstorming	0.915	0.320
Classification	0.921	0.478
Closed QA	0.924	0.388
Creative Writing	0.908	0.304
General QA	0.922	0.329
Info Extraction	0.923	0.433
Open QA	0.925	0.401
Summarization	0.918	0.403

Table 8: Quantitative assessment of the synthetic prompt paraphrasing task using BERT score and Rouge-L scores for each class.

C.2 Qualitative Examples

Low Quality : answering

reference: What are some things I could do as a tourist near Honolulu?

prediction: Explore popular attractions and activities in and around Honolulu, such as visiting Pearl Harbor, hiking Diamond Head, or relaxing on Waikiki Beach.

BERT score: 0.638 | **Rouge-L:** 0.059

reference: What do you do as a player playing RollerCoaster Tycoon 2?

prediction: Design and manage your own theme park, making decisions on ride placement, staff hiring, and customer satisfaction.

BERT score: 0.605 | **Rouge-L:** 0.0

reference: what is a dog?

prediction: Canine species, typically characterized by their furry coats, wagging tails, and ability to be domesticated.

BERT score: 0.594 | **Rouge-L:** 0.0

Low Quality : missing context

reference: classify the following as good stuff vs bad stuff:

inflation

wars

collaboration

renewable energy

soil erosion

peace

harmony

violence

bloodshed

prediction: Distinguish between positive and negative concepts:

BERT score: 0.613 | **Rouge-L:** 0.0

Moderate Quality : Useful, but verbose and over-specific

reference: Why should I visit Netherlands?

prediction: What are the top attractions and experiences that make the Netherlands a must-visit destination?

BERT score: 0.704 | **Rouge-L:** 0.1

reference: Please extract the characteristics of a desert from this passage and list in bullet format.

prediction: Identify and summarize the key features of a desert as described in the passage, presenting them in a concise bullet-point list.

BERT score: 0.742 | **Rouge-L:** 0.378

reference: Tell me about StarCraft.

prediction: What is the background and gameplay of the popular real-time strategy video game StarCraft?

BERT score: 0.727 | **Rouge-L:** 0.105

reference: What is a Wonton?

prediction: What are wontons, and what do they typically consist of?

BERT score: 0.756 | **Rouge-L:** 0.143

reference: What is the history of coffee?

prediction: Explore the origins and evolution of coffee from its ancient discovery to its global popularity today.

BERT score: 0.718 | **Rouge-L:** 0.273

High Quality : high Rouge-L

reference: What are the ingredients of a classic Greek salad?

prediction: What are the typical components of a traditional Greek salad?

BERT score: 0.925 | **Rouge-L:** 0.737

reference: What is the Canada's national sport?

prediction: What sport is officially recognized as Canada's national sport?

BERT score: 0.847 | **Rouge-L:** 0.706

reference: What is the name of Google's email service?

prediction: What is the name of Google's popular email platform?

BERT score: 0.969 | **Rouge-L:** 0.842

High Quality : moderate Rouge-L

reference: What is the best way to answer an interview question?

prediction: How can I effectively respond to an interview question to make a positive impression?

BERT score: 0.801 | **Rouge-L:** 0.333

reference: What principles does conversion rate optimization share with direct response marketing?

prediction: What similarities exist between the methodologies of conversion rate optimization and direct response marketing?

BERT score: 0.847 | **Rouge-L:** 0.56

reference: Classify these as things you would want to wear for a rainy day or a very hot day: umbrella, ball cap, rain jacket, t-shirt, sunglasses, rain boots, flip-flops

prediction: Categorize these items as suitable for a rainy day or a hot day: umbrella, rain jacket, rain boots, ball cap, sunglasses, t-shirt, flip-flops.

BERT score: 0.919 | **Rouge-L:** 0.618

High Quality : low Rouge-L

reference: Why are visualisations useful for communicating information?

prediction: What benefits do visualizations offer in conveying complex data effectively?

BERT score: 0.836 | **Rouge-L:** 0.0

reference: Which sport is similar to baseball?

prediction: What game shares similarities with baseball in terms of gameplay and rules?

BERT score: 0.815 | **Rouge-L:** 0.111

D Appendix: Training

D.1 FLAN-T5-XL

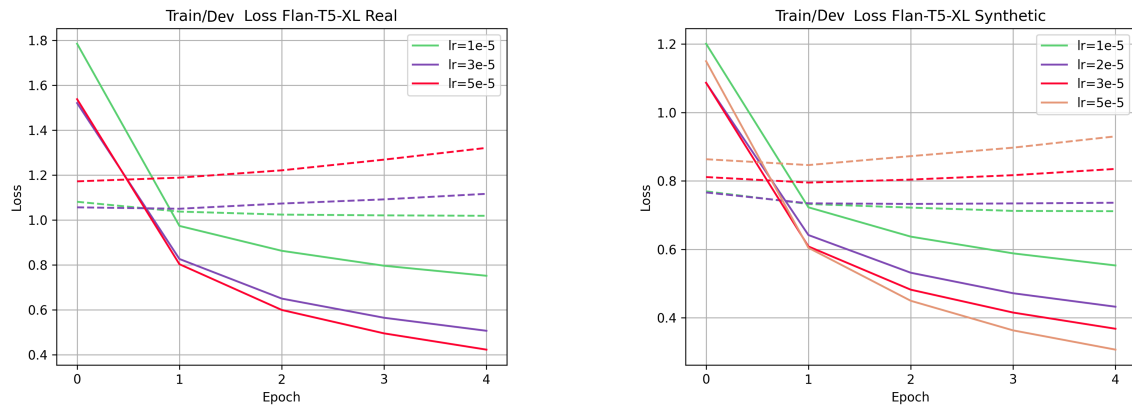


Figure 5: Training and development loss (dashed) for FLAN-T5-XL with the designated learning rates. FLAN-T5-XL-R (left) and FLAN-T5-XL-S (right).

D.2 Llama-8B-Instruct

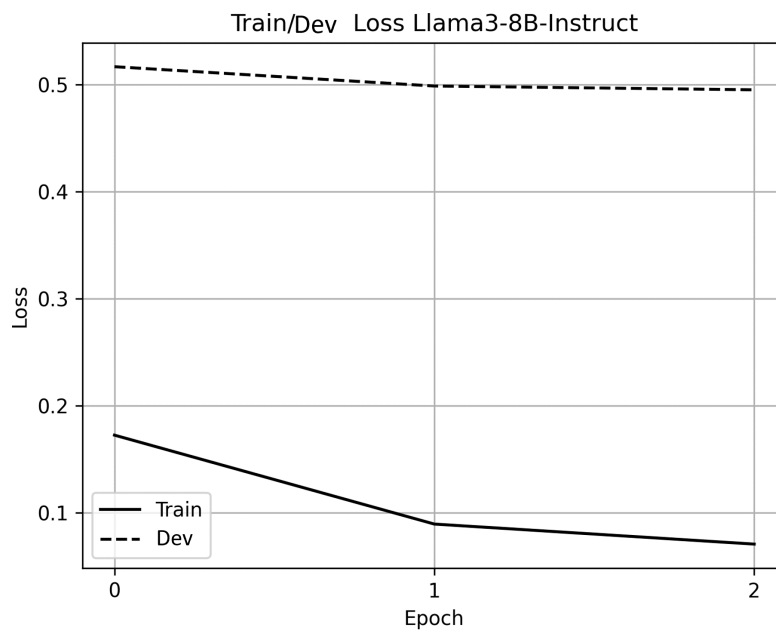


Figure 6: Training and development loss for Llama3-8B-Instruct.

D.3 Testing Dropout - Sequence Classification

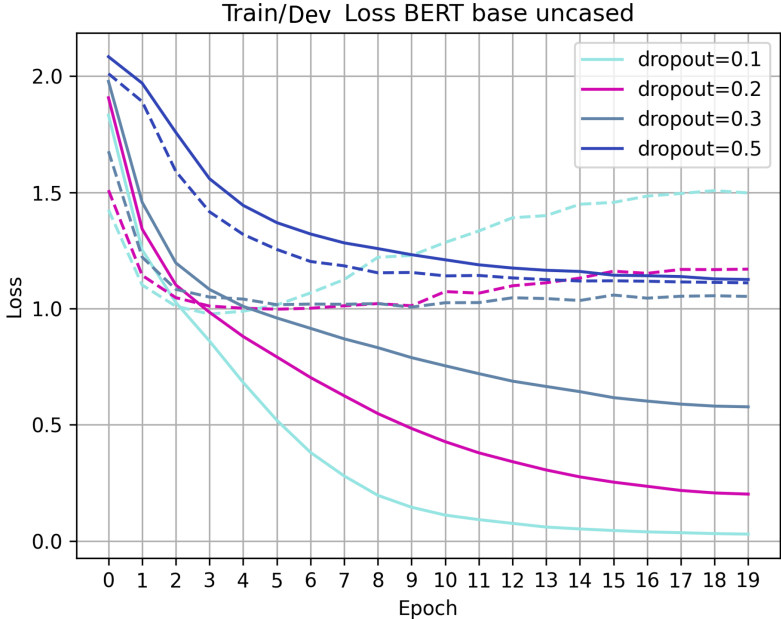


Figure 7: Training and development loss (dashed) for the four different dropout rates for the BERT-classifier. The lowest observed development loss before overfitting occurs at epoch 3 with dropout=0.1.

E Appendix: Paraphrasing Issues for Short Concise Prompts

Summarization

Direct Answer

Reference: What is Chess?

Prediction: A two-player strategy board game where players move pieces to capture the opponent's king.

Partial Answer

Reference: What is Attack on Titan?

Prediction: Can you tell me about the popular Japanese manga and anime series that takes place in a world where humans are under attack by giant humanoid creatures?

Verbose and Complex

Reference: What was the Byzantine Empire?

Prediction: What was the historical empire that succeeded the Roman Empire and existed from the 4th to the 15th century, primarily in the region of modern-day Greece, Turkey, and Eastern Europe?

General QA

Direct Answer

Reference: What is knitting?

Prediction: Knitting is the process of creating fabric by manipulating yarn with needles or other tools to create rows of interconnected loops.

Partial Answer

Reference: What is non dual philosophy?

Prediction: What is the concept of non-dualism, which posits that reality is fundamentally unified and lacks distinctions between subject and object, self and other, or consciousness and the external world?

Verbose and Complex

Reference: What is hot pot?

Prediction: What is the concept of a communal, simmering broth-based dish where diners cook their own ingredients at the table?

Open QA

Direct Answer

Reference: What is comparative genomics?

Prediction: Comparative genomics is the study of the similarities and differences between multiple genomes to identify evolutionary relationships, functional conservation, and genetic variations.

Partial Answer

Reference: What is DeFi?

Prediction: What does the term "Decentralized Finance" (DeFi) refer to?

Verbose and Complex

Reference: Who were the Sumerians?

Prediction: What was the identity of the ancient civilization that inhabited Mesopotamia and is credited with developing writing, governance, and architecture?

Closed QA

Direct Answer

Reference: What do you do as a player playing RollerCoaster Tycoon 2?

Prediction: Design and manage your own theme park, making decisions on ride placement, staff hiring, and customer satisfaction.

Partial Answer

Reference: What is Ontology?

Prediction: What is the study of the nature of existence, including the relationships between concepts and entities?

Verbose and Complex

Reference: What is Autism and how varied is the spectrum?

Prediction: What are the defining characteristics of Autism Spectrum Disorder (ASD) and how does it manifest in individuals with varying degrees of severity and symptom presentation?

Information Extraction

Direct Answer

Reference: What is The Silmarillion?

Prediction: A detailed description of J.R.R. Tolkien's collection of stories and legends about the ancient history of Middle-earth.

Partial Answer

Reference: What is Snow White and the Seven Dwarfs?

Prediction: What is the classic fairy tale about a princess named Snow White who takes refuge with a group of seven small miners, the dwarfs, to escape the wrath of her evil stepmother?

Verbose and Complex

Reference: What is the central limit theorem

Prediction: Can the average of a large sample of independent and identically distributed random variables be approximated by a normal distribution, regardless of the underlying distribution of the variables?

F Appendix: Prediction Quality Assessment

Strict Constraint

generalQA

reference: What programming language should I learn?

prediction: What programming language should I learn?

BERT-score: 1.0 | Rouge-L: 1.0

reference: What is the most popular sport in the UK?

prediction: What is the most popular sport in the UK?

BERT-score: 1.0 | Rouge-L: 1.0

OpenQA

reference: What is Bar Method?

prediction: What is Bar Method?

BERT-score: 1.0 | Rouge-L: 1.0

reference: What is the difference between benefits and compensation?

prediction: What is the difference between benefits and compensation?

BERT-score: 1.0 | Rouge-L: 1.0

Summarization

reference: What is Penguin Books?

prediction: What is Penguin Books?

BERT-score: 1.0 | Rouge-L: 1.0

reference: Who was Robert Ross Munro?

prediction: Who was Robert Ross Munro?

BERT-score: 1.0 | Rouge-L: 1.0

Brainstorming

reference: Are there more wheels or doors in New York City?

prediction: Are there more wheels or doors in New York City?

BERT-score: 1.0 | Rouge-L: 1.0

reference: Who are the greatest hockey players of all time?

prediction: Who are the greatest hockey players of all time?

BERT-score: 1.0 | Rouge-L: 1.0

Information Extraction

reference: What is a fishing vessel?

prediction: What is a fishing vessel?

BERT-score: 1.0 | Rouge-L: 1.0

reference: When was George Washington born?

prediction: When was George Washington born?

BERT-score: 1.0 | Rouge-L: 1.0

ClosedQA

reference: What is a PFD?

prediction: What is a PFD?

BERT-score: 1.0 | Rouge-L: 1.0

reference: What is the Pareto principle?
prediction: What is the Pareto principle?
BERT-score: 1.0 | Rouge-L: 1.0

Classification

reference: Identify which instrument is string or percussion: Octoban, Socavon
prediction: Identify which instrument is string or percussion: Octoban, Socavon
BERT-score: 1.0 | Rouge-L: 1.0

reference: Identify which instrument is string or percussion: Kirikoketa, Ichigenkin
prediction: Identify which instrument is string or percussion: Kirikoketa, Ichigenkin
BERT-score: 1.0 | Rouge-L: 1.0

Moderate Constraint

generalQA

reference: Why do humans like dogs?
prediction: Why do people like dogs?
BERT-score: 0.975 | Rouge-L: 0.8

reference: How do I get to Juneau, Alaska?
prediction: How do you get to Juneau, Alaska?
BERT-score: 0.981 | Rouge-L: 0.857

OpenQA

reference: What is the paleolithic?
prediction: What is Paleolithic?
BERT-score: 0.953 | Rouge-L: 0.857

reference: Who is John D. Rockefeller?
prediction: Who is John Rockefeller?
BERT-score: 0.955 | Rouge-L: 0.889

Summarization

reference: Where is the annual indoor rowing World Championship?
prediction: Where is the indoor rowing world championship held?
BERT-score: 0.948 | Rouge-L: 0.875

reference: Who were the major players in the Watergate conspiracy?
prediction: Who were the key players in the Watergate conspiracy?
BERT-score: 0.986 | Rouge-L: 0.889

Brainstorming

reference: List the five French mother sauces used in classic french cuisine.
prediction: What are the five French mother sauces used in classic French cuisine?
BERT-score: 0.862 | Rouge-L: 0.87

reference: What are some places to visit in Anchorage?
prediction: What are some good places to eat in Anchorage?
BERT-score: 0.962 | Rouge-L: 0.824

Information Extraction

reference: How many games did Edwin play in the NBA?
prediction: How many games did Edwin played in the NBA?
BERT-score: 0.985 | Rouge-L: 0.889

reference: What were the original four federal holidays in the United States?
prediction: What are the original four federal holidays in the United States?
BERT-score: 0.989 | Rouge-L: 0.909

ClosedQA

reference: Where in Australia is Palm Cove located?
prediction: Where is Palm Cove located?
BERT-score: 0.935 | Rouge-L: 0.833

reference: Given this paragraph, what is the strongest part of the scapholunate ligament?
prediction: What is the strongest part of the scapholunate ligament?
BERT-score: 0.932 | Rouge-L: 0.857

Classification

reference: Which characters belong to DC or Marvel Universe? Atom, Hank Pym
prediction: Identify which characters belong to DC or Marvel Universe: Hank Pym, Atom
BERT-score: 0.862 | Rouge-L: 0.87

reference: Classify each of the following music genres as broadly popular or niche: Rock, Hip Hop, Pop, Heavy Metal, Bluegrass, Country
prediction: Classify each of the following as either "broadly popular" or "niche": Rock, Hip Hop, Pop, Country, Bluegrass
BERT-score: 0.9 | Rouge-L: 0.811

Lenient Constraint

generalQA

reference: Which products apple sell?
prediction: What kind of products does Apple sell?
BERT-score: 0.803 | Rouge-L: 0.545

reference: What are submersibles for?
prediction: What is a submersible?
BERT-score: 0.907 | Rouge-L: 0.25

OpenQA

reference: What is an earphone?
prediction: What are earphones?
BERT-score: 0.94 | Rouge-L: 0.286

reference: Why do people prefer Windows over Linux?
prediction: What is the difference between Linux and Windows?
BERT-score: 0.801 | Rouge-L: 0.133

Summarization

reference: How long have musical instruments been in existence?

prediction: What was the first musical instrument?

BERT-score: 0.801 | Rouge-L: 0.143

reference: Why are cats so picky about what they eat?

prediction: What do cats like to eat?

BERT-score: 0.819 | Rouge-L: 0.267

Brainstorming

reference: Is it better to rebuild or remodel a house?

prediction: Should I build or remodel my house?

BERT-score: 0.853 | Rouge-L: 0.375

reference: I have a small ball, what games can I play?

prediction: What sports can you play with a small ball?

BERT-score: 0.847 | Rouge-L: 0.316

Information Extraction

reference: What are the names of the main islands of Hawaii in alphabetical order?

prediction: What are the names of the islands in the Hawaiian Islands?

BERT-score: 0.874 | Rouge-L: 0.64

reference: What was the Ghazi Attack in Indian subcontinent?

prediction: What was the Ghazi submarine?

BERT-score: 0.877 | Rouge-L: 0.615

ClosedQA

reference: Who composed the theme song for the movie Marvin's Room?

prediction: Who wrote and performed Marvin's room?

BERT-score: 0.804 | Rouge-L: 0.444

reference: Which is the most popular Italian Song Festival

prediction: What is the most famous music festival in Italy?

BERT-score: 0.814 | Rouge-L: 0.471

Classification

reference: Tell me which continent the following cities are in:

- Seattle
- Helsinki_
- Lagos
- Osaka
- Istanbul

prediction: Tell me whether these cities are located in North America, Europe, Africa, Asia or neither:
Seattle, Helsinki, Lagos, Osaka, Istanbul

BERT-score: 0.812 | Rouge-L: 0.588

reference: Classify each of the following music

genres as broadly popular or niche: Rock, Hip Hop, Pop,
Heavy Metal, Bluegrass, Country
prediction: Classify each of the following
as either "broadly popular" or "niche":
Rock, Hip Hop, Pop, Country, Bluegrass
BERT-score: 0.9 | Rouge-L: 0.811

Uncovering Hidden Intentions: Exploring Prompt Recovery for Deeper Insights into Generated Texts

Louis Give Timo Zaoral Maria Antonietta Bruno

University of Oslo

{louisgi,timoz,marabru}@uio.no

Abstract

Today, the detection of AI-generated content is receiving more and more attention. Our idea is to go beyond detection and try to recover the prompt used to generate a text. This paper, to the best of our knowledge, introduces the first investigation in this particular domain without a closed set of tasks. Our goal is to study if this approach is promising. We experiment with zero-shot and few-shot in-context learning but also with LoRA fine-tuning. After that, we evaluate the benefits of using a semi-synthetic dataset. For this first study, we limit ourselves to text generated by a single model. The results show that it is possible to recover the original prompt with a reasonable degree of accuracy.

1 Introduction

The rapid evolution of Natural Language Generation (NLG) in creating human-like text has introduced new challenges within the field of NLP. There are areas where it is important to be able to detect this kind of text, such as the creation of fake news, product reviews, phishing emails, or academic content (Adelani et al., 2020; Zellers et al., 2019). Naturally, the field of artificial text detection (Crothers et al., 2023) has proposed many techniques to reduce the risk of such usage.

While this field has seen significant advancements, recovering the original prompt remains an underexplored area. Indeed, it would help us to better understand the vast corpus of texts generated by language models and the underlying intentions of their creators. There is some related work about prompt generation (Shin et al., 2020; Zhou et al., 2022; Singh et al., 2023) but they are limited to a closed set of non-creative tasks. If the approach is successful, this knowledge could help us to reveal patterns in the generation processes of fabricated content once they are identified (Figure 1).

This paper introduces initial work in this area, employing techniques such as zero-shot and few-



Figure 1: Potential usage

shot learning (Wei et al., 2022), but also Low-Rank Adaptation (LoRA) fine-tuning (Hu et al., 2022), to explore the feasibility of recovering the initial prompts. Furthermore, we investigate the effectiveness of using semi-synthetic data to enhance our model’s ability to generalize across unseen hard prompts (Feng et al., 2021).

Our experiments focus on text generated by a single model, setting a controlled framework for the initial exploration. The findings reveal a promising potential for prompt recovery, suggesting pathways for further research on generalization. This initial study lays the groundwork for future investigations that could expand to text generated from many models and more diverse data scenarios, ultimately aiming to improve the interpretability of the intention behind AI-generated text and their traceability.

2020) to be more representative of online data.

The dataset is split as follows: 80% train, 10% validation, and 10% test.

The primary limitation of this preliminary study is that we use only one model to generate responses. If this initial approach is found to be promising, it would be interesting to investigate the generalization of this approach to get closer to the data available on the internet. Some samples will be shown in the qualitative analysis (Figure 6).

3.2 Synthetic Instructions

In order to improve the robustness of our model, we experiment with synthetic prompt generation (Feng et al., 2021). We take inspiration from SELF-INSTRUCT (Wang et al., 2023) and the Stanford Alpaca project (Taori et al., 2023) where they fine-tuned an LLM on instructions and responses generated by another LLM.

We focus on “creative writing” prompts, the most complex category, and the closest to online data. Creative writing presents a challenge for prompt prediction due to its complexity, high variability, evaluation difficulties, and inherent subjectivity (Figure 2b).

We generate around 3,000 instructions (Figure 4) with following setup:

- *Model*: Mistral-7B-Instruct
- *Prompt*: “You are asked to come up with a set of 20 creative task instructions. These task instructions will be given to a GPT model and we will evaluate the GPT model for completing the instructions. You can write something like that: "Write a poem inspired by the colors of a sunset" or "Write a short story about a character who can communicate with animals" or "Create a news about the difficulty of finding housing in San Francisco" or "Adress a letter to my mom to convince her that I should be able to get a cat." or "Generate a scholarly abstract on the impact of climate change on agriculture from a global perspective.””
- *Temperature*: 1.5
- *Top_p*: 0.9
- *Top_k*: 200

The response to these instructions is then created in the same way as for human prompts (Figure 3). They are only added to the train set.

We now have a so-called semi-synthetic dataset.

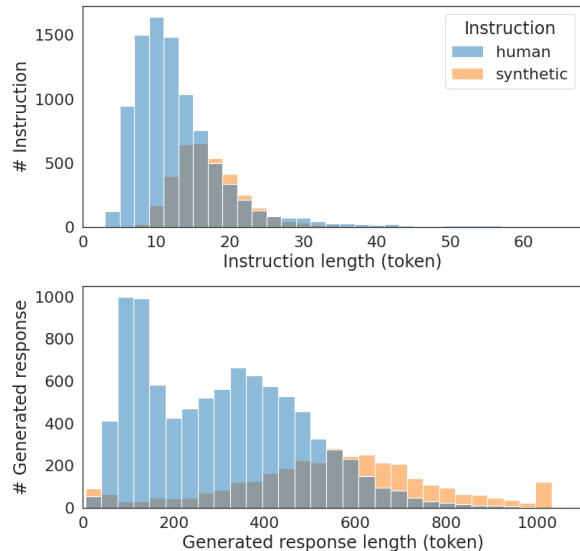


Figure 4: Length distribution of the instructions and generated responses

4 Experimental Results

4.1 Zero-shot and Few-shot Learning

Setup

To create a baseline, we carry out zero-shot and few-shot in-context learning as our first experiments. These are both performed with the *Mistral-7B-Instruct* model. On the one hand, this allows us to create a comparison for our later fine-tuned models, and, on the other hand, we can also test how the model is fundamentally suitable for the task of prompt prediction.

The first step in implementing zero-shot and few-shot learning is to set up a suitable prompt. The right choice of words and a clear task for the model must be taken into account. All in all, we create the following two prompts which are then both used individually for the experiments:

1. "`<s>`[INST] What prompt was used to generate this Text using LLM?
Text: {generatedText}
Prompt: [/`INST`]"
2. "`<s>`[INST] Predict and return only the prompt which was used to generate the Text.
Text: {generatedText}
Prompt: [/`INST`]"

With the two prompts presented, we have a question as a task with prompt n^o1 and a request with prompt n^o2. This gives us some variation in the experiments. We also use a low temperature of

Method	Metric	Category					Average (balanced)
		Brainstorming	Creative Writing	General QA	Open QA	Summarization	
Zero-shot	ROUGE-L	0.28	0.32	0.29	0.31	0.28	0.30
	MiniLM similarity	0.67	0.69	0.69	0.71	0.71	0.70
	BERTScore	0.96	0.96	0.96	0.96	0.96	0.95
Few-shot	ROUGE-L	0.38	0.37	0.50	0.48	0.37	0.42
	MiniLM similarity	0.80	0.74	0.84	0.83	0.76	0.79
	BERTScore	0.96	0.96	0.96	0.96	0.96	0.96
LoRA	ROUGE-L	0.45	0.40	0.50	0.57	0.44	0.47
	MiniLM similarity	0.82	0.75	0.83	0.84	0.81	0.81
	BERTScore	0.96	0.97	0.97	0.97	0.97	0.97
LoRA w/ synthetic data	ROUGE-L	0.47	0.43	0.56	0.58	0.46	0.50
	MiniLM similarity	0.83	0.78	0.87	0.85	0.82	0.83
	BERTScore	0.96	0.97	0.97	0.97	0.96	0.97

Table 1: Prompt recovery quantitative metrics (higher is better) on the human test set

Prompt	Method	ROUGE-L	MiniLM	BERTScore
1	zero-shot	0.18	0.54	0.95
1	three-shot	0.37	0.75	0.96
2	zero-shot	0.30	0.70	0.95
2	three-shot	0.42	0.79	0.96

Table 2: Comparison of both prompts between zero-shot and three-shot with ROUGE-L and BERTScore.

0.4 to reduce repetitions. This setting means that everything is ready for the zero-shot experiments.

For the few-shot in-context learning, three examples still have to be selected from our data set. These are then given to the model as a template for processing the task. In the final prompt, the examples are placed as follows:

```
"<s>[INST] Predict and return only
the prompt which was used to
generate the Text.
Text: {sampleText1}
Prompt: {samplePrompt1}
Text: {sampleText2}
Prompt: {samplePrompt2}
Text: {sampleText3}
Prompt: {samplePrompt3}
Text: {generatedText}
Prompt: [/INST]"
```

Results

We evaluate zero-shot and few-shot performance for each instruction of the test set (900 human instructions) (Table 2). We can see that prompt n^o2 performs better in every metric. In addition, it can be seen for both prompts that, as expected, three-shot learning represents a significant improvement compared to zero-shot learning. In both cases, the ROUGE-L score is almost doubled.

Category	Method	Qualitative Score	
		Prompt n ^o 1	Prompt n ^o 2
Brainstorming	zero-shot	2.0	2.2
	three-shot	1.9	2.3
Creative Writing	zero-shot	1.7	1.7
	three-shot	2.0	2.0
General QA	zero-shot	2.2	2.3
	three-shot	2.5	3.1
Open QA	zero-shot	1.8	2.1
	three-shot	2.3	2.6
Summarization	zero-shot	2.0	1.9
	three-shot	1.9	2.5

Table 3: Comparison of both prompts between zero-shot and three-shot based on qualitative analysis. Scale:

- 4 – Perfect instruction
- 3 – Correct instruction with minor imperfections
- 2 – Valid instruction with errors
- 1 – Irrelevant or invalid

Since we cannot interpret in detail from the ROUGE-L and BERTScore to what extent the prediction is satisfying, we conduct a qualitative analysis of 200 examples in total (Table 3). It confirms the findings obtained previously with the other three metrics. On the one hand, a significant improvement can be seen when using three-shot learning, and, on the other hand, the performance of prompt n^o2 is also slightly better.

Overall, it seems that the model is able to predict a similar instruction. What is clearly recognizable is that in categories such as General QA and Open QA, the model also shows better performance, probably because they are easier. This is also confirmed by the quantitative metrics in the Table 1. As we can see, the “creative writing” cate-

gory performs the worst, which again confirms our intention to generate synthetic data of this type.

4.2 Fine-tuning

Setup Once we have obtained correct results with our baseline, we fine-tune the model using LoRA (Hu et al., 2022), a parameter-efficient fine-tuning method. We follow the recommendations of the authors by setting $r = 32$ and $\alpha = 64$. Now we have “only” 85 million parameters to train compared to the 7 billion from *Mistral-7B*, our frozen backbone. We apply the same prompt as in the zero-shot experiment to limit the total runtime. This prompt is also masked out during back-propagation so that we don’t train the model to predict it. Training takes 45 minutes for 3 epochs with an NVIDIA A100.

Results We make the following observations from the results in Table 1. The use of LoRA brings a 12% increase in ROUGE-L and 3% in our sentence similarity metric compared to the best few-shot technique. This success underscores the potential of parameter-efficient fine-tuning techniques like LoRA in enhancing the capabilities of large language models without the extensive costs typically associated with full model fine-tuning.

4.3 Adding Synthetic Data

Setup To improve our results we experiment with the addition of synthetic prompts, more precisely in the worst-performing category (creative writing) because some others are already really good (Table 1). The generation process is described in section 3. We use the same fine-tuning technique (4.2).

Results The addition of synthetic data leads to create our best-performing model with a new improvement of metrics. Table 1 shows a increase of 6% for ROUGE-L and 2% in MiniLM similarity. Figure 5 and 6 present the key point of this study: the low number of bad-quality predictions.

4.4 Qualitative Analysis

The best model shows a significant improvement with the metrics used, almost doubling ROUGE-L over the zero-shot baseline. To understand this performance on quantitative metrics, we also analyzed qualitatively the results obtained in 4.3, as we did with the zero and three-shot in 4.1.

Global analysis The Figure 6 shows the distribution of the scores awarded for the individual exper-

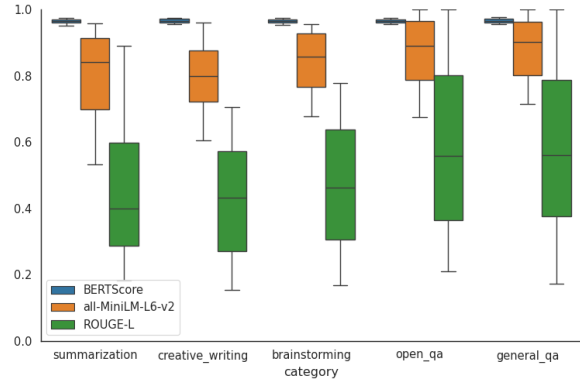


Figure 5: Fine-tuning performance following the category with semi-synthetic data

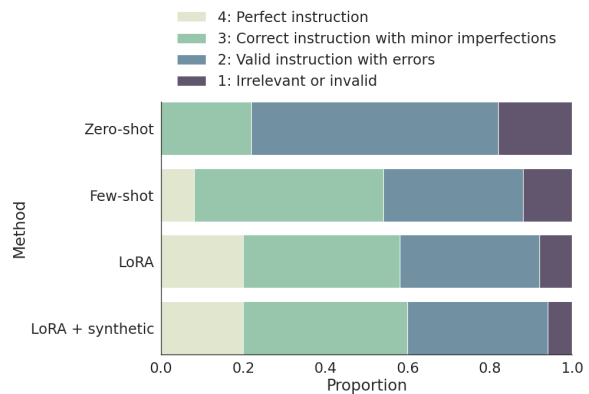


Figure 6: Qualitative analysis. 50 annotations for each method (10 for each category)

iments. It shows a similar improvement trend to the quantitative metrics. This can be seen starting from zero-shot, three-shot through fine-tuning, and finally to the best fine-tuning with the addition of synthetic data. In the last one, 60% of the prompts received a score of 3 or 4, which indicates a good to very good prediction whereas the invalid part is less than 5%, which confirms that it is possible to train a model to recover the original prompt.

Detailed analysis In the Table 4, we have listed examples of the original and the corresponding predicted prompts of our best solution. The top two prompts show two very good examples of how the model predicts the prompt.

The first example shows a perfect prediction of a slightly easier prompt.

The second example shows a slightly more difficult prompt, which the model also predicts almost perfectly. Only the sentence structure is slightly different and the Bernoulli principle itself is asked for, but this does not change the meaning of the prompt.

Generated Response (Input)	Predicted Prompt	Original Prompt	Qual. Score
Spokane, Washington does not have a widely recognized or commonly used nickname. The city's official nickname is "The Lilac City," which is derived from the large number of lilac bushes that were planted in the city in the late 1800s. However, this nickname is not as commonly used as some other city nicknames, and it may not be well-known outside of the Spokane area.	What is the nickname of Spokane, Washington?	What is the nickname for Spokane, Washington?	4
Bernoulli's principle is an empirical law that relates the pressure, velocity, and potential energy of a fluid in motion. It states that an increase in the velocity of a fluid in a tube causes a decrease in pressure, and vice versa, while the potential energy remains constant. This principle can be derived from Newton's second law of motion by considering the net force acting on a small element of fluid in a pipe. [...]	What is Bernoulli's principle and how can we derive it from Newton's second law of motion?	How can Bernoulli's principle be derived from Newton's second law of motion?	3
All twelve months have at least 28 days. However, only February in a normal year (a non-leap year) has exactly 28 days. All other months have more than 28 days. For example, March has 31 days, April has 30 days, and May through December each have at least 31 days. In a leap year, February has 29 days instead of 28.	What is the number of days in each month of a normal year?	How many months have 28 days?	1
Blue light therapy is a form of phototherapy that uses light-emitting diodes (LEDs) or cool white fluorescent lamps to emit blue light. People like blue light therapy for several reasons: 1. Treatment of Seasonal Affective Disorder (SAD) [...]	What is blue light therapy used for?	Why do people like Blue Light Therapy?	2
In the age of software and technology, personal privacy has become a hotly debated topic. With the increasing interconnectedness of our lives, the collection and use of personal data have become ubiquitous. On one hand, technology has brought about incredible conveniences and efficiencies, making our lives easier in countless ways. On the other hand, it has raised serious concerns about who has access to our personal information, how it is being used, and what the potential consequences might be. From a personal perspective , the loss of privacy can feel like an invasion of our most intimate spaces. Every time we use a search engine, make a purchase online, or engage in social media, we are leaving a digital footprint that can be traced and analyzed. [...]	Write a thoughtful and persuasive essay on the importance of personal privacy in the age of software and technology.	Write a point of view on personal privacy in the age of software and technology.	3

Table 4: Original and predicted instructions from our model fine-tuned on semi-synthetic data. (randomly selected)

The third and fourth prompts seem easy to predict at first glance, but they show how easily the models can still be misled. For example, the third example asks for specific months that have 28 days. However, as other months also appear in the generated text, this is not clear and so the prediction is promptly too inaccurate.

In the fourth example, the question of why and what is somewhat confused. This also occurs a few times when question words are mixed up and the context is slightly different.

In the last example, we present an example of a creative writing task. This shows how the model

predicts this prompt very well from a content perspective. However, the wording of the prompt does not completely match the original, but is it really possible if we use sampling during response generation?

Nevertheless, even the worst examples shown here are not a complete failure. The three positive examples (score ≥ 3) are representative of the 60% of how this prediction works and looks. (Figure 6).

No significant difference could be crystallized in the qualitative analysis between the fine-tuning approach on the human training data and the one enriched with synthetic data. However, this may

be due to the small number of examples analyzed. This could be investigated in more detail in a further study.

To summarize, the qualitative analysis identified some very good predictions based on the fine-tuning approaches. This confirms that a model can achieve an accuracy of at least 60% on a prompt recovery with a given generated text from one model.

5 Related Work

Instruction generation Our study is not the first work about generating prompts. A series of recent works (Shin et al., 2020; Zhou et al., 2022; Singh et al., 2023) cover the subject in a different manner with **autoprompting** not prompt recovering. It seeks the optimal prompt for a specified human-generated text formatted as “Input: [X] Output: [Y]”. Tasks typically include style transformations, sentiment analysis, basic logic, or language translation. However, this approach, restricted to a pre-defined set of non-creative tasks, does not apply to the varied nature of online-generated text.

Reverse engineering of models There is a growing area of interest in reverse engineering architecture and hyperparameters of models (Oh et al., 2018; Asnani et al., 2023). This improved knowledge of AI-generated content provides new descriptors for tracing them, which is in line with our final goal. It also reveals the limits of proprietary models.

Instruction-tuned LMs More generally, several studies (Ouyang et al., 2022; Wei et al., 2022; Wang et al., 2023) have demonstrated that base language models can successfully follow general language instructions when fine-tuned with "instructional" datasets. This explains why our solution works if the underlying LM has already been trained on this type of data.

Language models for data augmentation As detailed by this survey (Feng et al., 2021), data augmentation with LMs enhances the diversity and volume of training data, thereby improving model robustness and performance.

6 Conclusion

This study presents a preliminary experiment to assess the feasibility of backtracking from generated text to its original prompt.

The findings from our experiments demonstrate that it is indeed possible to recover the original prompts with a reasonable degree of accuracy.

While our results are promising, they are not without limitations. The scope of this study was confined to texts generated by a single model. Generalizing to multi-model generated texts seems to be the next step toward a real-world application.

References

- David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. 2020. [Generating sentiment-preserving fake online reviews using neural language models and their human-and machine-based detection](#). In *Advanced information networking and applications: Proceedings of the 34th international conference on advanced information networking and applications (AINA-2020)*, pages 1341–1354. Springer.
- Vishal Asnani, Xi Yin, Tal Hassner, and Xiaoming Liu. 2023. [Reverse engineering of generative models: Inferring model hyperparameters from generated images](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. [Free dolly: Introducing the world’s first truly open instruction-tuned llm](#).
- Evan Crothers, Nathalie Japkowicz, and Herna L Viktor. 2023. [Machine-generated text: A comprehensive survey of threat models and detection methods](#). *IEEE Access*.
- 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.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. [The curious case of neural text de-generation](#). In *International Conference on Learning Representations*.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. [Mistral 7b](#). *arXiv preprint arXiv:2310.06825*.
- 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.

- Seong Joon Oh, Max Augustin, Bernt Schiele, and Mario Fritz. 2018. [Towards reverse-engineering black-box neural networks](#). *International Conference on Learning Representations*.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. [Training language models to follow instructions with human feedback](#). *Advances in neural information processing systems*, 35:27730–27744.
- Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. [AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.
- Chandan Singh, John X. Morris, Jyoti Aneja, Alexander Rush, and Jianfeng Gao. 2023. [Explaining data patterns in natural language with language models](#). In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 31–55, Singapore. Association for Computational Linguistics.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. [Minilm: deep self-attention distillation for task-agnostic compression of pre-trained transformers](#). In *Proceedings of the 34th International Conference on Neural Information Processing Systems*.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. [Self-instruct: Aligning language models with self-generated instructions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. [Finetuned language models are zero-shot learners](#). In *International Conference on Learning Representations*.
- Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. [Defending against neural fake news](#). In *Advances in Neural Information Processing Systems* 32.
- 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*.
- Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. [Large language models are human-level prompt engineers](#). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.