Proceedings of the Fifth IN5550 Workshop on Neural Natural Language Processing (WNNLP 2023)

Andrey Kutuzov, David Samuel, Sondre Wold
and Erik Velldal (Editors)

June 06, 2023

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 Fifth IN5550 Teaching Workshop on Neural Natural Language Processing (WNNLP 2023). Spurred by great advancements in neural approaches to NLP, this is the fifth 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 NLP research tasks.

The workshop received ten submissions (by 17 authors), of which all have been accepted for publication as part of the WNNLP 2023 proceedings (this volume).

This programme would not have been possible without the assistance of all our reviewers, whose careful and constructive feedback has been an important element in finalising the individual contributions. To encourage the spirit of good peer review, we have made the decision to include an Outstanding Reviewer award in this workshop, in addition to the traditional Best Paper award. Further, to emphasise the ties this workshop has had to the IN5550 class, we have also made the decision to include an Outstanding Coursework award.

The Programme Committee has selected the paper *Guessing Menu ingredients with a Transformer: A Qualitative Analysis of the Training Corpus of a Norwegian-to-English Transformer model used for Menu Translations*, by Amir Basic, Cornelius Bencsik and Torstein Forseth, for the WNNLP 2023 Best Paper award, reflecting a combination of solid work with the data, in-depth error analysis and clever experimentation.

Among the pool of wonderful WNNLP 2023 reviewers, the Programme Committee finds that one deserves a special mention, for providing especially detailed and constructive feedback to their peers. The recipient of the WNNLP 2023 Outstanding Reviewer award is Cornelius Bencsik.

And finally, while the coursework that led to this workshop may seem like a distant memory, we also award the Outstanding Coursework awards to Amir Basic, Cornelius Bencsik and Torstein Forseth, who have received full marks on their deliveries this semester.

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

And, last - but not the least - warmest thanks to all participants of this workshop, who spent many a sleepless nights working on the projects that are certain to make WNNLP 2023 an exciting and stimulating event!

WNNLP 2023 Committee
Oslo; June 06, 2023

Programme Committee

Magnus Nytun
Oliver Getz Rodahl
Martins Gintovts
Anton Zelentsov
Erlend Kopperud
Sander Helgesen
Nolwenn Bernard
Torstein Forseth
Cornelius Bencsik
Amir Basic
Lilja Charlotte Storset
Fredrik Aas Andreassen
Roman Machacek
Narae Park
Jie Bian
Lu Xing
Lucas Georges Gabriel Charpentier
Sondre Wold
Egil Rønningstad
David Samuel
Andrey Kutuzov
Erik Velldal

Neural Machine Translation Chairs

David Samuel

Definition modeling

Andrey Kutuzov

Targeted Sentiment Analysis Chairs

Egil Rønningstad

General Chair

Erik Velldal

Table of Contents

Exploring Efficient Use of Domain-Based Data Samples in Targeted Sentiment Analysis .. 1
Magnus Nytun and Oliver Getz Rodahl

Exploring Cross-Domain Effects in Targeted Sentiment Analysis for Norwegian 7
Martins Gintovts

Neural Machine Translation: Small Transformer Models Are All You Need 17
Anton Zelentsov, Erlend Kopperud and Sander Helgesen

French-English Translation of Artwork Description: A Neural Machine Translation Study. 27
Nolwenn Bernard

Guessing Menu ingredients with a Transformer: A Qualitative Analysis of the Training Corpus of a Norwegian-to-English Transformer model used for Menu Translations 37
Amir Basic, Cornelius Bencsik and Torstein Forseth

Targeted Sentiment Analysis for Norwegian: An Experimental Study of Sentiment Intensity 45
Lilja Charlotte Storset and Fredrik Aas Andreassen

Benchmarking Targeted Sentiment Analysis Models 55
Roman Machacek

Definition Modeling for Chinese and Korean with Encoder-Decoder Language Models..... 65
Lu Xing and Narae Park

Make Less Become More: Explore Practical Approach to Nor-En Translation 77
Jie Bian

Leveraging monolingual encoders and decoders to perform Neural Machine Translation ... 87
Lucas Georges Gabriel Charpentier

Exploring Efficient Use of Domain-Based Data Samples in Targeted Sentiment Analysis

Magnus P. Nytun
magnuspn@uio.no

Oliver Getz Rodahl
olivegr@uio.no

Abstract

The Norwegian language model NorBERT3 was fine-tuned using an enriched $NoReC_{fine}$ dataset and evaluated for cross domain performance with the goal of estimating how useful this would be in a specific business application, namely using newly acquired reviews in an unseen category (domain) to make product related decisions based on Targeted Sentiment Analysis (TSA). Our fine-tuning results suggest that model selection and size is more important than adding new review samples to any specific data partition for maximizing an F1-score. However, there is a slight increase in performance when all samples are added to the training set, and that increase may be important for some applications. As such, data acquired for new review categories should exclusively be added to the training set to effectively leverage this new information. To confirm these results and evaluate how the amount of new reviews impact reliability, a test across domains was done with the model *NorBERT3-large*. We found that when using this model for inference without any fine-tuning on new domains, performance will vary significantly. As new reviews are added to the training set, the performance will stabilize and generally improve to a minor degree. Additional testing of scalability and using other models is encouraged to verify our results. In addition to NorBERT3 we also fine-tune a multilingual model, XLM-RoBERTa, on mixed Norwegian and English samples, with the expectation that less specialized models perform better on new data. Results are greatly dependent on seeding, and generally unstable.

1 Introduction

Targeted Sentiment Analysis (TSA) is a text mining problem within the field of natural language processing (NLP) that deals with opinions and emotion. As opposed to Sentiment Analysis (SA), in which a sentiment is extracted from an entire document (i.e. reviews, blog posts, social media posts), TSA attempts to answer the question of where those opinions or sentiments are directed. Persons, locations, organizations, objects, services, events, or anything else one can express opinions about are potential targets of SA (Liu, 2010a). This information can then be used in downstream tasks, for instance to do market research or use as directions to improve some specific aspects of a product.

We take the perspective of a business interested in doing TSA to measure the sentiment for a category of products. They have little to no labelled data for their own products in this category (in-domain), but have access to labelled data in other categories (out-of-domain). Over time, they are able to add more and more labelled in-domain data as it becomes available. To learn how best to utilize this newly accessible data, we measure the effect of gradually adding in-domain data to the development data and how model performance vary across categories given domains of different sizes. Can a business derive actionable knowledge from the results of TSA when they lack data in a given domain?

TSA problems can be approached as classification tasks (Liu, 2010b). It is closely related to Named Entity Recognition (NER), where entities (like targets in TSA) are identified in text. The basic format is IO-tagging, where words are classified as either inside (I) or outside (O) a named entity. A beginning (B) tag is commonly added to the list of classes (Lample et al., 2016). From this template, literature describe different variations depending on what is being researched. In our case, 5 tags are used to identify both sentiment and target (Table 1).

In Section 3 we start presenting our experiments, starting with searching for the most efficient data split. Here we utilize both Norwegian and multilingual large language models. We also introduce a low effort English TSA dataset to our training data. In Section 4 our work with incremental addi-

tion of in-domain samples is presented, followed by measuring the impact of domain sizes on model performance in Section 5. Lastly, our work is concluded in Section 6.

2 Data

We run our experiments and evaluations on the $NoReC_{fine}$ dataset introduced by (Øvrelid et al., 2020). This dataset consists of almost 8000 sentences which are carefully reviewed by roughly 300 reviewers. The data is originally sourced from Norwegian Review Corpus (Velldal et al., 2018). $NoReC_{fine}$ contains information about the intensity of each sentiment, but we disregard this part of the dataset for not being directly related to our goals. Our version of $NoReC_{fine}$ data consists of reviews where each token in the documents are labelled as shown in Table 1. The sentences are carefully labelled following a prescribed method, a procedure which is described in (Øvrelid et al., 2020). An accompanying metadata file[1] adds additional information about each document, such as language, rating, source, or more importantly for our use: a category tag (also referred to as domain or genre).

Table 1: Labels in the $NoReC_{fine}$ dataset.

Label	Definition
"O"	Outside
"B-targ-Positive"	Beginning positive
"I-targ-Positive"	Inside Positive
"B-targ-Negative	Beginning negative
"I-targ-Negative"	Inside Negative

We are interested in measuring and optimizing cross-domain performance. Therefore, we enrich the $NoReC_{fine}$ with genres retrieved from NoReCs metadata. Each sentence belongs to one of the genres listed in Table 2.

Each document in a dataset is split by spaces, often at the word and punctuation level. The sentence "I fjor kom 3DS-spillet Donkey Kong Country Returns, og det var absolutt et stilig bekjentskap." is thus represented like so:

```
['I','fjor','kom','3DS- spillet
   ↪ ','Donkey','Kong','Country
   ↪ ','Returns',',','og','det','
   ↪ var','absolutt','et','
   ↪ stilig','bekjentskap','.']
```

[1] https://github.com/ltgoslo/norec

Table 2: Categories in the $NoReC_{fine}$ dataset.

Category	Count
screen	3807
music	2692
products	2181
literature	1089
games	767
stage	376
restaurants	340
sports	149
misc	36

3 Data Partitioning

What is the most efficient use of data when you have a limited amount of in-domain labelled samples? Should the in-domain samples be placed in the training, validation or be distributed in both? In this section we seek to answer these questions. We split the $NoReC_{fine}$ dataset into 5 subsets.

1. **Other train** (n=6099): This set contains all of genres except for the screen genre. The samples in this dataset are used for fine-tuning.

2. **Other validation** (n=1531): This set contains the same genres as the "other train" set, but is used to validate model performance during fine-tuning.

3. **Screen train** (n=1272): This set represents our limited amount of in-domain data. This data can be added to the training, validation or both datasets.

4. **Screen validation** (n=1272): This set is used to validate model performance for each model and split when fine-tuning is completed.

5. **Screen test** (n=2535): This is the test set. The models are not evaluated on this until the very end.

Screen is singled out and can be viewed as the in-domain genre. If we relate this to the story in Section 1 the business mentioned could be working with screen plays. This means they are interested in doing TSA for their screen plays, but have limited amount of labelled data. Screen is in fact the largest genre in the $NoReC_{fine}$ dataset as we can see from Table 2. This lessens the burden of working with a limited amount of samples, and we can cover more ground with a greater number of experiments with

more reliable results due to the possible differences in sample sizes and splits.

3.1 The NorBERT3 Family

We fine-tune four different NorBERT3 (Samuel et al., 2023) model sizes: xs, small, base and large. The models are fine-tuned with a batch size of 16 and a learning rate of 5e-5. We implement early stopping to reduce the chance of overfitting to the training data while reducing the time it takes for each model to fine-tune. We set the models to fine-tune for 200 epochs, but the fine-tuning is stopped before this because of early stopping. The fine-tuning is stopped if a model's macro F1-score hasn't improved by at least 0.001 for the previous five epochs. Fine-tuning is repeated three times for each size with different seeds, and the average macro F1-score is reported in the table below.

Table 3: Macro F1-score for various data splits and NorBERT3 sizes.

Model	All-in-Val	All-in-Train	50/50 Split
xs	0.40	0.43	**0.43**
small	0.45	**0.49**	0.48
base	0.49	**0.52**	0.50
large	0.52	**0.56**	0.54

From Table 3 couple of pattern reveals. Firstly, adding the in-domain samples to the training set yields the highest macro F1-score for NorBERT3 small, base and large. 50/50 split is marginally better than All-in-Train for NorBERT3 xs. The performance achieved using a 50/50 split is not much worse than using an All-in-Train split. Adding all new samples to the validation set has the weakest performance out of the various splits for all sizes of NorBERT3. Overall, the size of the model seems to be more important than where you put your in-domain samples for model performance. This of course comes at a cost of more computation and longer inference time.

3.2 XLM-RoBERTa

We conduct a similar experiment with XLM-RoBERTa large and base model (Conneau et al., 2019) to explore whether using a multilingual model, less specialized by definition, leads to a more robust model more adept to generalizing to unseen domains.

(Radford et al., 2022) shows that exchanging quantity for quality in audio datasets leads to a model more adept for zero-shot. Therefore, we produce a low effort English TSA dataset. The data is sourced from SemEval 2014 (Pontiki et al., 2014) and the dataset stems from subtask 4[2]. We used a version of the dataset collected from Kaggle[3]. The dataset is cleaned to resemble the structure of the NoReC dataset. We remove duplicate rows, keeping the first occurrence of each document. As the label schema is different from the NoReC dataset we remove documents that are labelled as *conflict* and replace the *neutral* tag with *O*. The final dataset consists of 1,951 samples of English restaurant reviews. Some examples are presented below.

> "But the staff was so horrible to us."
>
> "The price is reasonable although the service is poor."

Thereafter we fine-tune four XLM-RoBERTa models. XLM-RoBERTa base and large is fine-tuned with and without adding the English restaurant reviews to the training data. We call the models fine-tuned on both English and Norwegian samples "mixed". We repeat the fine-tuning three times with different seeds. Since the results are very unstable and highly dependent on what seed is used, the maximum macro F1-score achieved is reported in the table below.

Table 4: Macro F1-scores for XLM-RoBERTa.

Model	All-in-Val	All-in-Train	50/50 Split
large	**0.52**	0.52	0.00
base	0.45	**0.47**	0.46
large-mix	0.51	**0.54**	0.54
base-mix	0.44	**0.49**	0.44

The performance of the models are very unstable, especially for the models fine-tuned exclusively on Norwegian data. Many of the fine-tuning runs results in a model that is unable to learn or overfits to the validation data resulting in a macro F1-score of 0. The fine-tuning becomes less prone to overfitting or not learning when the English low effort dataset is introduced in the training data. This also increases the model performance, and the XLM-RoBERTa mixed large model shows comparable performance to the NorBERT3 large model. While this comparison is not completely fair, given that

[2]https://alt.qcri.org/semeval2014/task4/#

[3]https://www.kaggle.com/datasets/charitarth/semeval-2014-task-4-aspectbasedsentimentanalysis

averages are reported for the NorBERT3 family and highest scores are reported for XLM-RoBERTa, it provides a decent performance estimate when this is kept in mind. Further research should be done to explore the effect of introducing more multilingual data to the training set.

4 Single-Domain Performance: Incremental Addition

In this section we explore the effect of incrementally adding in-domain samples to the various data splits. We use the same hyperparameters from section 3. We select the strongest model from section 3, which is NorBERT3 large.

We fine-tune the NorBERT3 large with 0, 5, 100, 200, 500, 1000 and 2535 samples of the in-domain category added to its respective split. This is done for the all in train split, all in validation split and when we add to both sets. For the both split the data is sliced based on the middle index, and the first half is distributed to the training set. The second half is distributed to the validation set. The data is partitioned in the same manner as in Section 3, but we have concatenated screen train and screen validation. This yields a dataset with 2,544 samples from the screen category. The samples from screen train and screen validation are distributed to the different splits: train, validation and both. The models are tested on the screen test set after fine-tuning. The fine-tuning is repeated three times with different seeds, and the average macro F1-score is reported in Figure 1.

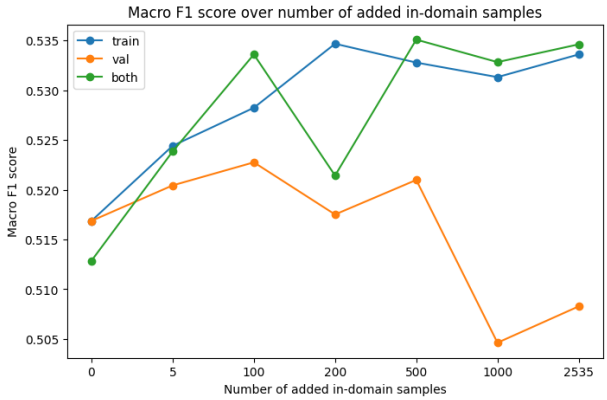


Figure 1: Incremental addition of in-domain samples

The macro F1-score is steadily increasing for the various splits until 200 samples are added to the datasets. When 200 samples are added to validation or distributed to both train and validation sets, the F1-score suddenly drops. When the samples are added to the training split the macro F1-score peaks. Since the same in-domain samples are added to the data for the same increments in fine-tuning, the sudden spike and pit at 200 samples might be explained by the nature of the samples that are being added. We speculate that when the samples are added to the validation data they might cause early stopping to seize fine-tuning because the validation score is not improving. When the same samples are added to the training split, they offer valuable learning which leads to higher performance. In the following increment, the samples are distributed to the training set for the train and both split. This means the model will make predictions based on these samples. We see that the "both" split increases drastically from the previous increment. This indicates that the content of the samples added might play a larger role than how many samples are being added.

The macro F1-score is gradually improving when we introduce more in-domain samples to the training and the both split. The same is not true when all the samples are added to the validation set. Early stopping might be partially to blame for this. But exclusively adding samples to the validation data does not seem like the most efficient use of new in-domain samples. The most efficient distribution is perhaps somewhere between adding everything to train and the 50/50 to train and validation.

5 Cross-Domain Performance: Impact of Domain Sizes

Following results from the previous experiments, we assume that adding all in-domain samples to the training set is the best course of action, but how much in-domain data do we really need in order to draw conclusions? Each category will likely contain different words and have targets/entities not present in other categories. From the outset, we therefore expect performance to drop when the model is presented with a category unseen during fine-tuning.

As we have seen in Table 2, there are vast differences in the amount of samples per domain in the dataset. On the lower end, 'misc' includes a measly 36 documents. On the upper end, 'screen' includes 3807 documents in total. This imbalance is likely similar to real-world scenarios, and an opportunity to challenge our expectations.

We executed the previous experiment on a single domain. In the following experiment, we repeated the fine-tuning and testing process on every other domain to learn if domain size would influence results.

5.1 The Domain Size Test

As NorBERT3-large achieved the best performance for the earlier experiments, only this model was used for the following tests. We commenced fine-tuning using the same batch size and learning rate as before, but we only used a single seed.

Our process was as follows for each category:

1. We isolated a category, removing it from the data set.
2. We split the remaining data set into train, validation, and test sets. This set was stratified by domain.
3. We added the isolated category back to the train and test sets 3 times (50/50, 25/75, and 0/100). The validation set was omitted from this process due to results from the previous experiment.

This process resulted in $categories \times n\_splits$ data sets (27), i.e. 3 different splits for each of the 9 categories in the $NoReC_{fine}$ dataset. Unsurprisingly, performance is better when there is more in-domain data in the training set. As can be seen in the bottom row of Figure 2, the model reaches a test F1-score of 50%-60% regardless of size. Our results indicate that the relative size of each category has little impact on performance, at least when the number of out-of-domain categories are 8, and that the model will generalize fairly well to new categories in this case. Whether these results scale to better performing systems, or if performance is more variable when the number of out-of-domain categories is lower, remains to be tested.

One drawback of having few in-domain samples, according to our data, is that it takes longer for the domain to stabilize. As seen in the upper row of Figure 2, learning is more volatile during the first handful of epochs when the number of in-domain samples in the training set is low. The model must fine-tune for longer to get reliable results, and greater care must be taken to stop fine-tuning at an appropriate time. Despite volatile initial fine-tuning for smaller sets of in-domain samples, there does not seem to be any link between the in-domain size and the time it takes for fine-tuning to converge. Using these methods will not negatively impact fine-tuning time.

While we find it unlikely to be the case that there is minimal differences in language across 9 categories, this is something that should be investigated further. It could be that the language used in each category is very similar, considering that they are all reviews, which may influence our results.

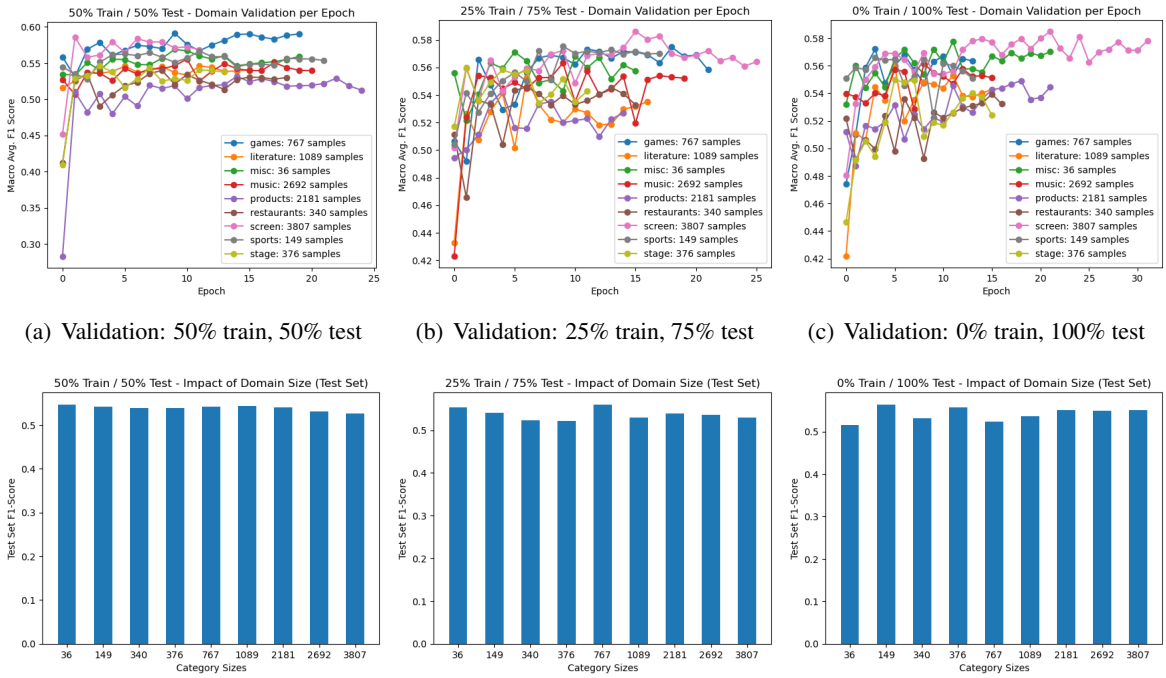
6 Conclusions

Our results clearly indicate that the most efficient distribution of in-domain samples is to the training set with some proportion in the validation set. What model is used is more important for model performance than in what data split you distribute your in-domain samples. With our results in mind we can finally conclude on the topic whether size matters or not - as our results indicate the larger the better.

XLM-RoBERTa also shows some promise when Norwegian samples are mixed with low effort English samples. Further exploring this would be interesting, and creating a dataset consisting of more multilingual TSA samples could perhaps produce models that are even more robust across domains. Evaluating whether our results generalize by fine-tuning other models on similar datasets is also encouraged.

Incrementally adding in-domain samples to train, validation or both data splits effects model performance. Especially when we going from a zero-shot to a few-shot learning scenario. The nature of the samples also seems to have a great impact on model performance. Therefore, the saying the more the merrier might not always be right when working with in and out-of-domain datasets.

Further results suggest that not a lot of in-domain data is needed to make use of $NoReC_{fine}$ and NorBERT3 for targeted sentiment analysis. There will be initial negative effects on performance, when the number of new samples are low and during early fine-tuning, but these effects may be negligible—depending on the current task. It may be useful to continuously analyze reviews on products in new categories as they become available.



(a) Validation: 50% train, 50% test (b) Validation: 25% train, 75% test (c) Validation: 0% train, 100% test

(d) Test: 50% train, 50% test (e) Test: 25% train, 75% test (f) Test: 0% train, 100% test

Figure 2: Top Row: Validation scores throughout the fine-tuning sequence for 50/50, 25/75, and 0/100 train-test splits. Bottom Row: The test score related to the size of each domain for each split.

References

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.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 263, San Diego, California. Association for Computational Linguistics.

Bing Liu. 2010a. Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing*, pages 2–3.

Bing Liu. 2010b. Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing*, page 10.

Lilja Øvrelid, Petter Mæhlum, Jeremy Barnes, and Erik Velldal. 2020. A fine-grained sentiment dataset for Norwegian. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5025–5033, Marseille, France. European Language Resources Association.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision.

David Samuel, Andrey Kutuzov, Samia Touileb, Erik Velldal, Lilja Øvrelid, Egil Rønningstad, Elina Sigdel, and Anna Sergeevna Palatkina. 2023. Norbench – a benchmark for norwegian language models. In *The 24rd Nordic Conference on Computational Linguistics*.

Erik Velldal, Lilja Øvrelid, Eivind Alexander Bergem, Cathrine Stadsnes, Samia Touileb, and Fredrik Jørgensen. 2018. NoReC: The Norwegian review corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Exploring Cross-Domain Effects in Targeted Sentiment Analysis for Norwegian

Martin Gintovt
martigin@ifi.uio.no

Abstract

Targeted Sentiment Analysis is the task of identifying a target of an opinion and classifying its corresponding polarity in a given text. This paper analyzes the effects of training neural models for that task while utilizing data for training and testing from (partly) heterogeneous domains or categories. The results demonstrate that models trained on the "screen" domain, containing reviews about movies and TV series show adequate performance when tested on other domains, while some categories are inherently difficult. Additionally, the results illustrate that combining data from various domains yields enhanced performance.

1 Introduction

Sentiment Analysis (SA), also referred to as Opinion Mining is a Natural Language Processing (NLP) task, where the goal is to determine an opinion behind a text unit and classify its sentiment (e.g. positive or negative). Research in SA comprises various domains, such as analyzing movie reviews (Pang et al., 2002). While SA would define an overall sentiment of a review, Targeted Sentiment Analysis (TSA) would provide entity-level sentiment for a specific target entity. Given an example movie review "*Effects were great, but plot horrendous"*, the entities would be *effects* and *plot*, each assigned positive and negative sentiments, respectively. TSA is a subset of SA and is a more fine-grained task. One of the major limitations of the usefulness of SA models, in general, is domain barriers. It is plausible to assume that a model trained on data originating from one domain (e.g. movie reviews) would demonstrate inferior performance when predicting sentiment in a different domain (e.g. sports).

This paper's aim is to investigate the cross-domain effects in the task of TSA for Norwegian. More specifically, it focuses on $\text{NoReC}_{\text{TSA}}$ dataset, exploring what domains appeared to be the most challenging to classify, how various combinations of data from heterogeneous domains affected the model's performance, as well as analyzes errors from particular domains.

The rest of the paper is structured as follows: Section 2 presents the data used for experiments. A description of the model selected for testing is given in Section 3. Section 4 surveys the experiments conducted. Section 5 summarizes and sets forth the findings and results of experiments. Error analysis is given in Section 6. Section 7 concludes the findings, before the paper is finished with Section 8, which concerns suggestions for future work.

2 Data

This paper makes use of $\text{NoReC}_{\text{TSA}}$, a dataset for fine-grained sentiment analysis in Norwegian. It is derived from $\text{NoReC}_{\text{fine}}$ dataset (Øvrelid et al., 2020), which in turn comprises a subset of the Norwegian Review Corpus (Velldal et al., 2018). The text units in the parent $\text{NoReC}_{\text{fine}}$ dataset are annotated with not only binary polarity labels (positive/negative) but also intensity labels (slight, standard, strong), making the dataset fitting for both binary and more fine-grained, 6-way classification. The underlying texts originate from professionally authored reviews from multiple news sources, including a variety of domains: 'literature', 'screen', 'sports', 'music', 'games', 'products', 'stage', 'restaurants', and 'miscellaneous'.

The data in $\text{NoReC}_{\text{TSA}}$ comes in conll-format, and includes a total of 5 labels, represented as **BIO**-labels with additional polarity: **B-targ-Positive** (B-Pos, beginning of the positive entity sequence), **I-targ-Positive** (I-Pos, inside positive entity sequence), **B-targ-Negative** (B-Neg, beginning of the negative entity sequence), **I-targ-Negative** (I-Neg, inside negative entity sequence), and **O** (outside). Figure 1 illustrates the data spread in the combination of train, development, and test splits of the dataset, whereas Table 1 demonstrates an

Token	Label
Ocarina	B-Pos
of	I-Pos
Time	I-Pos
kalles	O
ofte	O
tidenes	O
beste	O
spill	O
.	O

Table 1: Example sentence from $\text{NoReC}_{\text{TSA}}$ 'games' domain. *Ocarina of Time* is an entity, labeled with positive sentiment. English translation of the sentence is as follows: '*Ocarina of Time is often called the best game of all time.*'.

Domain	Sentences
Screen	3807
Music	2692
Literature	1089
Products	2181
Games	767
Restaurants	340
Stage	376
Sports	149
Misc	36

Table 2: Number of sentences in each domain of $\text{NoReC}_{\text{TSA}}$ dataset: test, development, and train splits combined.

example word sequence with its respective labels extracted from the 'games' domain. Additionally, the number of sentences per domain (train, development, and test data combined) can be found in 2. The data can be obtained in the GitHub repository of the IN5550 course[1].

3 Models

For the task of TSA this paper employs BERT (Bidirectional Encoder Representations from Transformers) pre-trained language model (LM), introduced by (Devlin et al., 2019). More specifically, it made use of NorBERT_3 (Samuel et al., 2023b): large pre-trained contextualized LM for Norwegian, based on BERT architecture. The pre-training phase of NorBERT_3 encompassed various text collections: Norwegian Wikipedia dumps both for Bokmål and Nynorsk from October 2022, Public domain texts released by the National Library of Norway in 2015[2], Norwegian News Corpus[3], Norwegian Colossal Corpus[4], as well as Norwegian part of of web-crawled mC4 corpus (Xue et al., 2021). Moreover, the authors employed the masked language modeling approach for pre-training NorBERT_3 and followed the optimized training method from (Samuel et al., 2023a). This approach differs from the standard BERT training.

In the same manner, as BERT, NorBERT_3 comes in several versions, including models with various numbers of parameters. This paper conducts an experiment where several variants of NorBERT_3 are tested on the $\text{NoReC}_{\text{fine}}$ dataset mentioned in Section 2. A thorough description of the experiment and its results are provided in Section 4 and Section 5, respectively. The results of the experiment motivate the use of the $\text{NorBERT}_{3,\text{base}}$ model, which is the main model utilized in the following experiments. The model comprises 123M parameters, 12 hidden layers, 12 attention heads, and hidden dimensions of size 786.

There is no motivation to choose this specific model for experiments - they solely require a model, powerful enough and trained on Norwegian texts. The ultimate goal is to observe how the performance differs based on the domains, not achieve state-of-the-art result.

4 Experimentation

Several experiments are carried out in this paper, where each one is conducted using the same model hyperparameters: batches of size **16**, **8** training epochs, a learning rate of **3e-5**, fine-tuning all of the models' parameters. The motivation behind the hyperparameters choice boils down to the goal of longer training with a lower learning rate. While a higher learning rate would possibly lead to a faster convergence, the aim is to see how the model performs over a longer span of time.

The first experiment described in subsection 4.1 concerns the selection of the NorBERT_3 model for further analysis. Subsection 4.2 surveys how using different domains, as well as their combination affects model performance. An investigation of the impact of adding a fixed number of samples

[1] https://github.uio.no/in5550/2023/tree/main/exam/tsa

[2] https://www.nb.no/sprakbanken/en/resource-catalogue/oai-nb-no-sbr-34/

[3] https://www.nb.no/sprakbanken/ressurskatalog/oai-nb-no-sbr-4/

[4] https://huggingface.co/datasets/NbAiLab/NCC

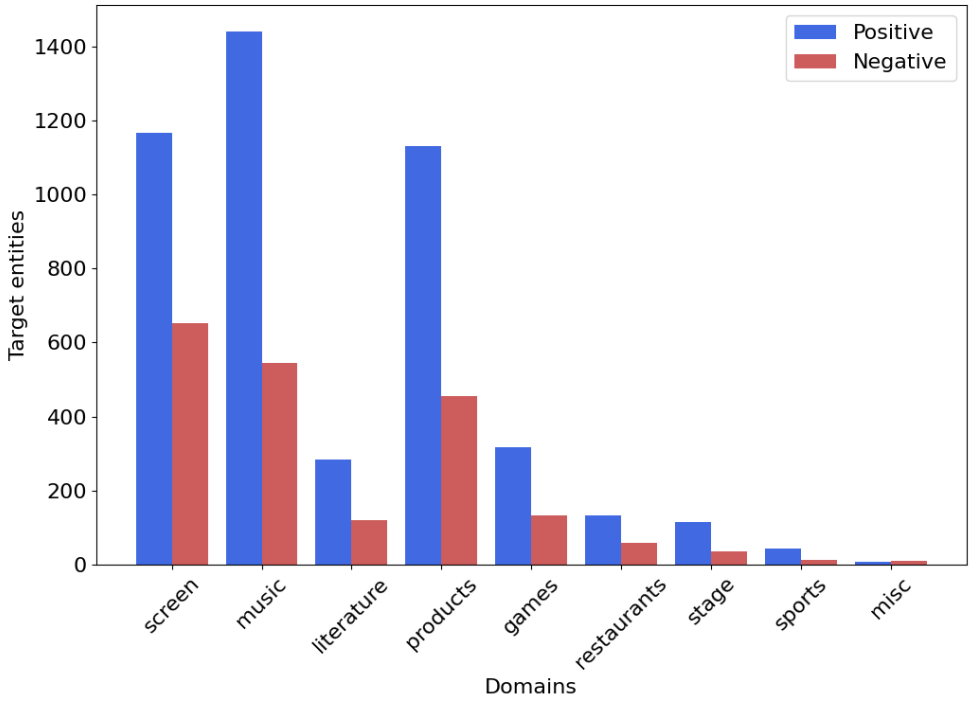


Figure 1: Spread of positively and negatively labeled target entities across domains in $\text{NoReC}_{\text{TSA}}$ data: train, development, and test splits combined. The plot was derived by counting the presence of B-Pos and B-Neg tags inside the tag sequence for each sentence, across all domains.

from target domains to training data is presented in subsection 4.3. The final experiment with adding different ratios of data from different domains is described in subsection 4.4.

4.1 Model selection

As mentioned in Section 3, several versions of NorBERT_3 are present at the moment of writing of this paper. The goal of the experiment is to determine the best-performing model to use in further analysis. In particular, 4 different versions of the model are tested: $\text{NorBERT}_{3,\text{small}}$, $\text{NorBERT}_{3,\text{base}}$, $\text{NorBERT}_{3,\text{large}}$, and $\text{NorBERT}_{3,\text{wiki-base}}$. A baseline is established for each model by training it on the train split, while assessing performance on the test split, provided by $\text{NoReC}_{\text{TSA}}$ dataset. Both splits comprise a combination of samples from nearly all domains, except 'sports' and 'misc'. These domains are therefore excluded in this experiment. Consequently, the most efficient and best-performing model is picked for further experiments. Finally, the experiment investigates the performance of the model on each separate domain from the test split.

4.2 Merging and training on different domains

The second experiment comes down to testing whether using data originating only from one or several different domains would potentially prove useful. The 'screen' and 'products' domains are of particular interest. The first one comprises the largest number of samples, whereas the latter, as described by the authors of the $\text{NoReC}_{\text{fine}}$ dataset, is perhaps the most diverse category, which comprises product reviews across a number of sub-categories. Consequently, it is intriguing to see whether such a domain will function well as the only source of training data. To conduct this and further experiments, all three data splits are merged together. Furthermore, 80% of the data from train domains is utilized for fine-tuning, while the remaining 20% is used for the purposes of testing. This is done in order to see how well will the model perform when trained and tested on data from the same domain ('screen' and 'products' in this particular case). The reason for merging the data is to increase the total amount of training and testing samples.

Furthermore, it seems intuitively plausible that combining data from different domains and using that data for training could potentially increase the

model's performance. Consequently, another experiment is conducted where the 'screen' and 'product' domains are merged together and used for training the model, while data from the remaining domains are utilized for testing purposes. Whereas the screen domain contains the most amount of data, the product domain serves as the most "universal" category.

It is important to note that during inference in the current and all subsequent experiments, a different amount of sentences is utilized in each domain. Specifically, each domain is tested using all the samples in that domain. The reasoning behind this decision boils down to some domains comprising a relatively low number of samples (e.g. sports). Utilizing an equal amount of samples would imply using only a small number of samples for testing in each domain, while there is an interest to see how the models will behave when tested on a broader amount of data.

4.3 Adding a fixed number of data from target domain to training data

In this experiment, the aim is to investigate the effect of adding a fixed amount of samples from the target domains to the training data. In particular, two tests are conducted: adding 10 and 100 samples. Furthermore, it is determined to drop the misc category in the testing phase, as it contains way too few samples for reliable performance assessment. This can be observed in Table 2.

4.4 Adding ratio of data from all domain to training data

This experiment focuses on testing whether adding a certain ratio of data from each category (depending on the total number of samples in that category) to training data would prove useful. More specifically, 3 different ratios are considered: taking **0.1**, **0.2**, and **0.3** of total data in each domain and adding it to the training data, while utilizing the rest for testing. Additionally, 3 different random seeds are employed for each run, ensuring some randomness when sampling data for training, and resulting in 9 runs in total. Consequently, the experiment results end up being more reliable, owing to the fact that the testing subset id slightly different between each run.

All the results for the aforementioned experiments are presented in Section 5.

5 Results

This section gives a report and surveys the results of experiments described in Section 4.

5.1 Experimenting with different NorBERT$_3$ models

Results for baseline testing can be found in Table 3. The metric utilized for assessing performance is F1-score, based on the Batista algorithm[5]. The results show that NorBERT$_{3,\text{large}}$ outperformed all of the models, while NorBERT$_{3,\text{wiki-base}}$ showed the lowest score.

As illustrated in Figure 2, NorBERT$_{3,\text{base}}$ and NorBERT$_{3,\text{large}}$ mostly demonstrate the highest F1-scores during tests in other domains. While the latter model tends to yield better performance in domains like 'literature', 'music', and 'products', NorBERT$_{3,\text{base}}$ outperforms it in 'restaurants' and 'stage' domains.

Given the fact that NorBERT$_{3,\text{base}}$ contains sufficiently fewer parameters than its expanded version NorBERT$_{3,\text{large}}$ (123M vs. 353M) while yielding similar performance, the choice of the model for further experiments ends on the first one. Moreover, the NoReC$_{\text{TSA}}$ is a dataset of arguably moderate size, which motivates the use of a less parameterized model to avoid any potential side effects of overfitting.

5.2 Investigating the use of various single and merged domains as training data

The results for testing the effect of using the 'screen' and 'products' domains as training data, as well as merging together them together are illustrated in Table 4. Despite the fact that the 'products' domain is perhaps the most diverse one, the model fine-tuned on it don't manage to perform significantly better than the same model fine-tuned on data from the 'screen' domain. While the model shows better performance in the 'games' domain (a **0.0284** increase) and the 'restaurants' domain (a **0.034** increase), it underperforms in other domains by a quite large margin. For instance, the performance in both 'sports' and 'misc' domains drops by **0.2027** and **0.2936**, respectively. This could potentially be due to a few reasons: (1) the 'products' domain contained fewer samples than the 'screen' domain, and (2) the content in the 'screen' domain

[5] https://www.davidsbatista.net/blog/2018/05/09/Named_Entity_Evaluation/

Model	Test split	Screen	Music	Literature	Products	Games	Restaurants	Stage
$\text{NorBERT}_{3,\text{small}}$	0.4652	0.4565	0.4857	0.4054	0.4706	0.5819	0.4206	0.5889
$\text{NorBERT}_{3,\text{base}}$	0.4844	0.4719	0.5580	0.3857	0.5069	0.6197	**0.4946**	**0.6296**
$\text{NorBERT}_{3,\text{large}}$	**0.5288**	0.4752	**0.5598**	**0.4461**	**0.5740**	0.5763	0.4639	0.6111
$\text{NorBERT}_{3,\text{wiki-base}}$	0.4626	**0.5047**	0.4677	0.4022	0.4391	**0.6254**	0.3117	0.5278

Table 3: Baseline performance of 4 different NorBERT_3 models trained on train split, while tested on test split of $\text{NoReC}_{\text{TSA}}$ as well as different domains from the test split.

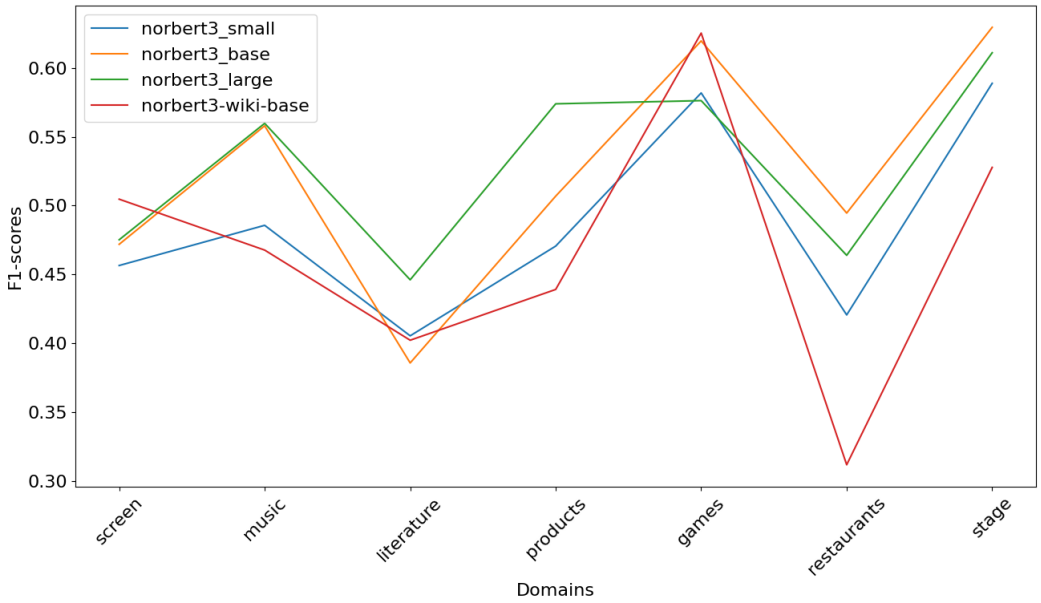


Figure 2: F1 scores achieved by each model trained on train split and tested on each domain from test split. $\text{NorBERT}_{3,\text{base}}$ and $\text{NorBERT}_{3,\text{large}}$ appear to be the most performant models.

Test domain	Screen	Products	Screen + Products
Screen	0.4842	0.3862	-
Music	0.4485	0.3946	**0.4584**
Literature	0.4158	0.3789	**0.4368**
Products	0.4515	0.5117	-
Games	0.4552	0.4836	**0.5195**
Restaurants	0.4213	0.4553	**0.4790**
Stage	**0.4602**	0.4024	0.4244
Sports	**0.2979**	0.0952	0.2925
Misc	0.5686	0.2750	**0.6627**

Table 4: F1-scores after training the model on 'screen', 'product' domains, and their combination. Intersection points where train and test domains are equal contain baseline results (using 80% of data for training and 20% for testing).

is inherently more similar to that of 'music, 'literature, and 'stage' domains.

Furthermore, the approach of merging together the 'screen' and 'product' domains improves the model's performance: except for the 'stage' and 'sports' domains, this approach allows to achieve superior performance in every other domain. For this reason, the combinations of these two categories are used as testing data for every following experiment.

5.3 Adding a fixed number of data from target domain

The results of the experiment found in Table 5 demonstrate improved performance in practically all domains. Moreover, adding 100 samples from the target category yields better performance in almost all domains (except for the 'stage' and 'sports' domains) when compared to only adding 10 samples or using data from the 'screen' and 'product' domains.

However, it appears that utilizing 100 samples degrades performance in the 'sports' domain quite significantly when compared to the effect of using 10 samples. Given that the 'sports' domain comprises a small amount of data, it is plausible to assume that after sampling 100 data instances, the few instances that are left (49) for testing differ from the sampled data.

5.4 Adding ratio of data from every domain

As described in Section 4, the results for this experiment are derived by running 3 experiments,

Test domain	Add-10	Add-100
Music	0.4588	**0.4616**
Literature	0.4313	**0.4457**
Games	0.5169	**0.5313**
Restaurants	0.4728	**0.5022**
Stage	**0.4788**	0.4453
Sports	**0.4189**	0.3353

Table 5: F1-scores after adding 10 or 100 samples from the target domain to the training data, initially consisting of data from 'screen' and 'product' categories.

Test domain	0.1 ratio	0.2 ratio	0.3 ratio
Music	0.4762	0.4738	**0.4771**
Literature	0.4291	0.4555	**0.4564**
Games	0.5195	0.5238	**0.5425**
Restaurants	0.4969	0.5057	**0.5134**
Stage	0.4859	**0.5011**	0.4973
Sports	0.3317	0.3879	**0.4362**

Table 6: F1-scores after taking a certain ratio of data from each domain, and adding it to the training data. Each run is conducted 3 times, using different seeds. The resulting scores are average of 3 runs.

utilizing various ratios of data (**0.1**, **0.2**, **0.3**) from every domain. Furthermore, each of the experiments is conducted 3 times, each time making use of a different random seed. This ensures that various samples from each domain are employed in training and testing. The final results, which are the average of 3 runs, are illustrated in Table 6. It appears, that utilizing 30% of samples from each domain proves the most beneficial for the model performance.

In general, it clearly seems that adding more data from each domain and allowing the model to fine-tune on it helps the model to yield greater performance. The model trained on a combination of only 'screen' and 'product' domains produces inferior results as opposed to the same model trained on data with an additional 30% of samples from each domain. For a more precise comparison, the latter model yields an increase in F1-scores equal to **0.0187** in the 'music' domain, **0.0196** in the 'literature' domain, **0.023** in the 'games' domain, **0.0344** in the 'restaurants domain', **0.0729** in the 'stage' domain and **0.1437** in the 'sports' domain.

6 Error Analysis

This section describes and analyses errors the model produces by means of inspecting a confusion matrix, in addition to investigating a number of sentences and their respective predictions and gold labels. Moreover, it inspects the most common tokens wrongly classified. In particular, it focuses on surveying the 'games' and 'sports' domains - the ones, where the model demonstrated the highest and lowest Batista F1 scores, respectively. All the results are derived from the epochs, where the model shows the highest score.

6.1 Confusion Matrices

Confusion matrices from the 'games' and 'sports' domains are illustrated in Figures 3 and 4, respectively.

As it can be observed, in the 'games' domain, the model tends to confuse the **B-Pos** labels with **O** labels and the other way around. This can be also observed for **B-Neg** tags but with a significantly lesser frequency. The exact same effect can be observed for **I-Pos** tags - in fact, the amount of misclassified target entities appears to be even higher. As with **B-Neg** tags, **I-neg** tags are more rarely misclassified when it comes down to confusing them with **O** tags. This effect could potentially be present due to the proportion of positive sentences in the 'games' being greater than that of negative sentences. Furthermore, the model gives an impression of doing an arguably good job at distinguishing between positive and negative sentences: not that many samples are misclassified. It is the prediction of the correct sentence span that is the most difficult to accomplish.

The same tendency can be observed in the 'sports' domain. The only major difference is that the numbers are significantly smaller. This is due to the fact that the 'sports' domain comprises a small number of samples.

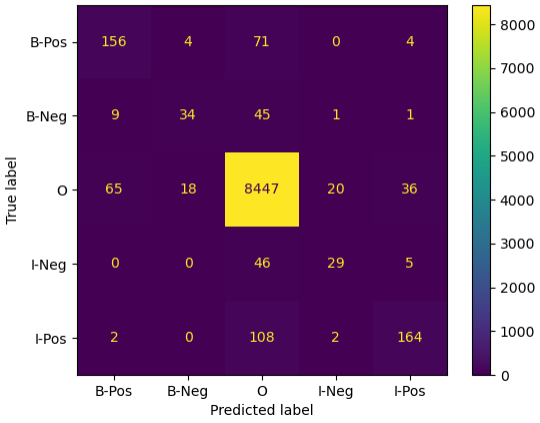


Figure 3: Confusion matrix from 'games' domain, evaluations derived from the epoch with the highest F1 score. The golden label is shown on the left, the predicted label is shown below the matrix. The diagonal of the matrix contains correctly classified target entities.

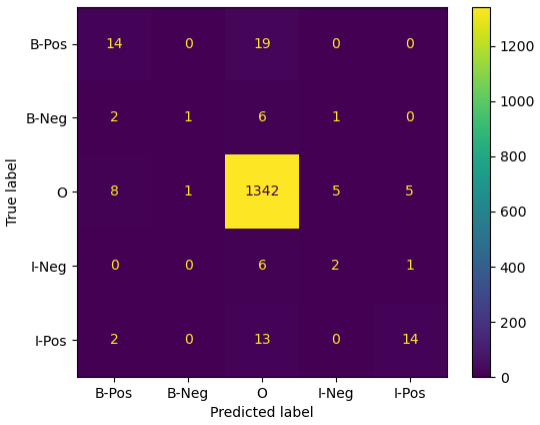


Figure 4: Confusion matrix from 'sports' domain, evaluations derived from the epoch with the highest F1 score. The golden label is shown on the left, the predicted label is shown below the matrix. The diagonal of the matrix contains correctly classified target entities.

6.2 Inspecting sentences

The paper further investigates specific sentences from both domains, their respective gold labels, and the model's predictions.

After observing a number of sentences, one could argue that the gold tags aren't necessarily faultless. A sentence from the 'games' domain with an incorrectly predicted span can be observed in Table 7. While '*Astrid Lindgres figurer (Astrid Lindgren's figures)*' and '*figurer (figures)*' are both

Token	Gold	Pred
Astrid	O	B-Pos
Lindgrens	O	I-Pos
figurer	B-Pos	I-Pos
gjør	O	O
seg	O	O
godt	O	O
i	O	O
et	O	O
barnespill	O	O
.	O	O

Table 7: Sentence from 'games' domain with incorrectly predicted entity spans. English translation of the sentence is as follows: '*Astrid Lindgren's figures do well in a children's game*'.

Token	Count	Gold	Pred
«	16	B-Neg	O
»	15	I-Neg	O
spillet	12	O	B-Pos
det	8	B-Pos	O
og	7	I-Pos	O
av	6	O	I-Pos
"	5	I-Pos	O
Det	5	O	B-Pos
spill	5	O	B-Pos
3	4	O	I-pos

Table 8: Top-10 wrongly classified tokens in 'games' domain

entities, the predicted one is arguably a more 'precise' entity than the gold entity. Moreover, the 'games' domain containes two following sentences: (1) '*Jeg elsker jobben min (I love my job)*' and (2) '*På den står det med store bokstaver at jeg elsker jobben min (It says in big letters that I love my job)*'. Whereas the first sentence includes gold positive entity '*jobben min (my job)*', that is not the case for the second sentence. It is debatable whether the second sentence should have had the entity omitted. Another example sentence that seems intriguing is '*Det er herlig, fengslende og vanedannende (It is wonderful, captivating, and addictive)*'. In this sentence, all the gold labels are equal to **O**, while the model predicted that the first word '*Det (It)*' is **B-Pos**. Again, it is probably plausible to assume that in the context of games, the word '*vanedannede (addictive)'* is of positive sentiment, indicating that the sentence could possibly contain a positive entity, exactly as the model predicted.

However, the model also demonstrates misclassifications caused by its ability to classify. An example is '*Bryce kan bytte mellom skytevåpen og sverd, et godt konsept som her dessverre har blitt miserabelt gjennomført (Bryce can switch between firearms and swords, a good concept that has unfortunately been miserably executed here)*'. Here, '*Bryce*' is given a positive polarity, whereas the gold label is negative and for the word *konsept (concept)*. Both the polarity and the span are not correct.

The 'sports' domain, while being moderate in size, simultaneously contains a fair amount of 'difficult' sentences. One such was '*Jone Samuelsen 3*'. The sentence represents a player, as well as its score. The first and last names are tagged with **B-Neg** and **I-Neg**, respectively, whereas the model predicted the same tags, but with positive polarity. It is arguably quite difficult to derive a correct prediction for this type of sentence since it is potentially ambiguous. An example of a sentence that model didn't manage to correctly classify due to its capabilities is '*- Et sensasjonelt mål av Götze! (- A sensational goal by Götze!)*'. Here, '*Götze*' is supposed to be positive, while the model didn't predict anything. This could potentially be attributed to the fact that the model have never seen the token '*Götze*' before.

Overall, the sentences in the 'sports' domain appear to be quite hard to predict correctly given their context.

6.3 Most common error words

The top 10 wrongly classified words for both 'games' and 'sports' domains are illustrated in Tables 8 and 9, respectively. Correlating to the findings from confusion matrices in 6.1, it can be observed from the tables that the most common source of error is defining the correct span of an entity. This can specifically be noticed in Table 8: quotation marks '«' '»' are the most commonly misclassified tokens. Moreover, the model seems to struggle to differentiate between all the other tags and **O** tags.

7 Conclusion

This paper analyzes the effects of conducting the task of TSA while utilizing different combinations of data from heterogenous domains, both for train-

Token	Count	Gold	Pred
Lionel	2	B-Pos	O
Messi	2	I-Pos	O
Manuel	2	O	I-Neg
Vi	2	B-Pos	O
Higuaín	2	O	B-Neg
Neuer	2	B-Neg	B-Pos
mot	2	B-Neg	O
Götze	2	B-Pos	O
laget	2	B-Pos	O
Estland	2	I-Neg	O

Table 9: Top-10 wrongly classified tokens in 'sports' domain

ing and testing. By making use of the $\text{NoReC}_{\text{fine}}$ dataset, and its various categories, while training and testing $\text{NorBERT}_{3,\text{base}}$ model, it was found that adding a bigger ratio of data from each domain to train data yield better performance in every single domain. The baseline was mostly never beaten, but that differed from domain to domain. Moreover, it was discovered that testing the model on 'games' produced the highest F1 scores, while the 'sports' category was the hardest to predict. Additionally, the error analysis in the paper demonstrated that the model was able to distinguish between positive and negative polarities, but struggled to predict correct entity spans. It also showed that some of the sentences' gold labels were potentially debatable.

8 Future work

Given the fact that $\text{NoReC}_{\text{fine}}$ comprises several domains, while only a few are more thoroughly tested in this paper, it would be intriguing to see how the combination of other domains would affect model performance. Moreover, a greater number of potentially more powerful models for Norwegian exist, offering possibilities to conduct these tests utilizing them. Additionally, it would be of great interest to implement Behavioral Testing (Ribeiro et al., 2020) for analyzing the effects of cross-domain TSA. Given the lack of data in certain domains, it would potentially prove beneficial to create short, simple sentences for each domain to investigate models' behavior.

References

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

Lilja Øvrelid, Petter Mæhlum, Jeremy Barnes, and Erik Velldal. 2020. A fine-grained sentiment dataset for Norwegian. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5025–5033, Marseille, France. European Language Resources Association.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

David Samuel, Andrey Kutuzov, Lilja Øvrelid, and Erik Velldal. 2023a. 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.

David Samuel, Andrey Kutuzov, Samia Touileb, Erik Velldal, Lilja Øvrelid, Egil Rønningstad, Elina Sigdel, and Anna Sergeevna Palatkina. 2023b. Norbench – a benchmark for norwegian language models. In *The 24rd Nordic Conference on Computational Linguistics*.

Erik Velldal, Lilja Øvrelid, Eivind Alexander Bergem, Cathrine Stadsnes, Samia Touileb, and Fredrik Jørgensen. 2018. NoReC: The Norwegian review corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

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

Neural Machine Translation: Small Transformer Models Are All You Need

Sander Helgesen
University of Oslo
sehelges@ifi.uio.no

Erlend Kopperud
University of Oslo
erlenmko@ifi.uio.no

Anton Zelentsov
University of Oslo
antonz@ifi.uio.no

Abstract

In this paper, we have examined the problem of neural machine translation. In particular, we consider the problem of drastically reducing the model size of neural machine translation models without sacrificing the overall performance. Furthermore, we experiment with various combinations and sizes of datasets as well as hyperparameter tuning of different models. For testing, we collected our own test dataset of 101 sentences from English to Norwegian, and achieve an average BLEU-score of up to 39.2 for one of our smaller models with less than 6 million trainable parameters. Through qualitative manual assessment, some of the translations are even better than what the BLEU-score suggests.

1 Introduction

In a world where cultures are getting closer than ever, surrounded by different languages, it is important we understand each other. The ability to be understood and communicate with people from different language backgrounds is important, yet challenging. This is where machine translation (MT) can potentially help bridge the gap between languages. In this paper, we train a neural machine translation (NMT) model using the transformers architecture (Vaswani et al., 2017) to translate sentences from English to Norwegian.

Neural machine translation models serve as valuable tools for communication across different languages. Conventional translation, i.e. human translation, takes time and requires bi- or multilingual knowledge, or the need of dictionaries. Having a MT model to translate sentences automatically can increase the usability and ease of access for translations. NMT are often more reliable and flexible models, due to their ability to learn context and predict human-like translations, compared to e.g. statistical MT models.

However, there is a big challenge in making these models accessible to a broader audience. One major issue is the substantial size and memory requirements of language models, which make it impractical for consumers to use them on their own devices. One could use e.g. a cloud service to host a NMT model and have these devices interface with this cloud model. This, however, requires the device to be connected to the internet to be able to use this model, and internet connections are not always reliable or available. Therefore, it is useful to develop small NMT models, that can be deployed to consumer devices without hogging data storage.

2 Background

2.1 Transformers

The transformer is a deep learning architecture first introduced by Google. (Vaswani et al., 2017) The idea behind its creation was to eliminate the recurrence (feedback loops in neural networks to maintain a state of memory between previous inputs and outputs) of previous state-of-the-art models within NLP, such as recurrent neural networks (RNNs) (Zaremba et al., 2014) and rely entirely on the attention mechanism (Vaswani et al., 2017). This was motivated by the fact that recurrent models took a long time to train as the sequence lengths of the inputs got larger. By utilizing the attention mechanism with global dependencies, training could be significantly more efficient with the possibility of the parallelization of the sequences.

The transformer is an encoder-decoder architecture. Its encoder creates an encoding of the input data that captures the meaning of the sequence, and the decoder receives the representation from the encoder and the input from the target data, and applies the attention mechanism to the two inputs. This is shown in connection between the left and right box in figure 1.

2.2 Attention Mechanism

In the transformer architecture typically two types of attention are used, self-attention and cross-

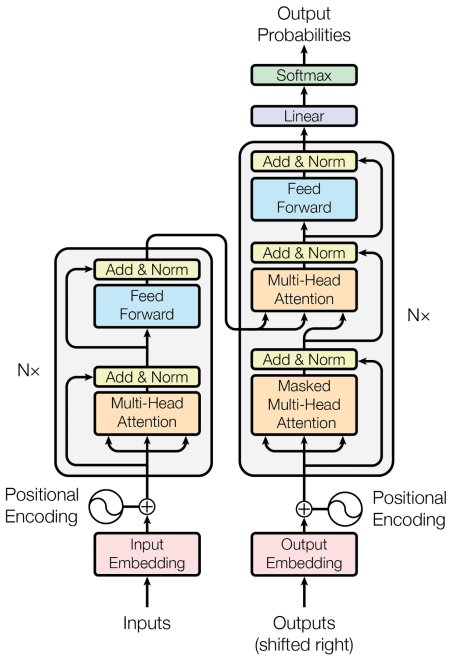


Figure 1: Transformer model architecture (Vaswani et al., 2017)

attention (Vaswani et al., 2017). In the encoder of the transformer, each layer uses the self-attention mechanism. It allows the model to attend to different parts of the input sequence while capturing dependencies between different positions. The decoder introduces an additional cross-attention mechanism after the self-attention mechanism. It allows the decoder to attend over the output of the encoder, capturing the relevant information needed for the decoding process.

Additionally, masked attention is used in the self-attention mechanism of the decoder stack to prevent the model from attending to future or unseen information, or padding tokens.

The attention mechanism of the transformer also uses multi-headed attention allowing the model to attend to different parts of the input sequence simultaneously, and capture different types of relationships. Instead of looking at only one projection of the inputs, the algorithm uses several variations of learned linear projections for the same sequence. Each attention head can focus on different input aspects, enabling the model to learn and incorporate diverse information. The outputs of multiple attention heads are then combined to form a comprehensive representation that enhances the model's ability to capture complex dependencies and improve its overall performance on various tasks.

2.3 Search

When training a machine translation model, a technique for generating meaningful outputs from probabilities for the possible next words in the sequence is needed. This technique is supposed to decode the output in a way such that it can be compared with the target sentence.

In this paper, we use two kinds of search algorithms in order to solve this problem. The first one is greedy search, in which only the token with the highest probability of being next is selected to generate the sequence. This is fast and easy to implement, but it is more likely to get stuck in a local optimum.

Another approach is to use beam-search, which evaluates the k highest probabilities and examines each path from each corresponding word until an end-of-sentence token is predicted. This is illustrated in figure 2. Beam search is more computationally heavy but is more likely to give better sequences, because it explores more of the probabilities for words. (Tillmann and Ney, 2003)

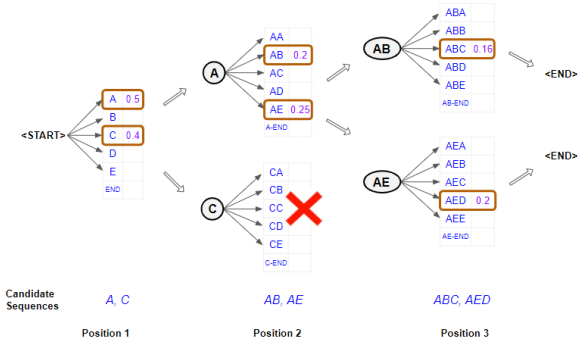


Figure 2: Illustration of the beam search algorithm. [1]

2.4 BLEU - SacreBLEU

Machine translation is both difficult and expensive to measure, as there is no single correct answer to a translation due to many different ways to phrase the same idea. However, there is a need to compare results during evaluation, therefore we want a general approach that can give a metric score. To achieve this, we use the SacreBLEU metric (Post, 2018), which is a variation of the well-known BLEU metric.

BLEU (Papineni et al., 2002) is a metric that takes a weighted geometric mean of n-grams for the machine-translated outputs compared to the target

[1] https://towardsdatascience.com/foundations-of-nlp-explained-visually-beam-search-how-it-works-1586b9849a24

sentences. The weighted geometric mean makes sure that longer sentences are weighted higher, as shorter ones are more likely to score higher by chance. This is due to the uni-gram implementation of BLEU, where shorter sentences have a higher chance of getting higher scores for n-grams with fewer correct words translated.

In this paper, we use SacreBLEU (Post, 2018) for evaluating the results of the translations. SacreBLEU is a more flexible version of BLEU that consider the different meanings punctuation can have. It also supports multiple reference translations when multiple valid translations are correct.

3 Method

3.1 Datasets

For this project, we are working with multiple bilingual datasets, consisting of sentence pairs with English and Norwegian sentences. The training datasets consist of sentences taken from government-owned websites and subtitles to movies.

The government dataset is made up of two publicly available corpora: The Bilingual English-Norwegian parallel corpus from the Office of the Auditor General (Riksrevisjonen) website [2], and Public Bokmål- English Parallel Corpus (PubBEPC) [3]. These corpora are sampled from four different government websites.

The subtitles dataset consists of unofficial subtitles to TV shows and movies submitted by users to OpenSubtitles, and formatted into a relatively large bilingual dataset by P. Lison and J. Tiedemann [4].

For validation during training we are using the book “Hound of the Baskervilles” by Arthur Conan Doyle, translated and formatted by Farkas Translations [5]. As of time of writing this website is down due to maintenance, but the book is a part of a collection called "Bilingual books".

The test dataset is our own constructed dataset with sentence pairs from Tatoeba [6], which is a collection of translated sentences for 420 different languages. We hand picked 100 arbitrary English-Norwegian sentence pairs of good quality, and added one sentence pair that we wrote ourselves, for a total of 101 data points.

The government dataset is considered to be of higher quality, as the translations are done by professionals to communicate important information over various government-owned websites. However, the language is quite formal and contains many specific glossaries about e.g. taxes, grants and specific organizations like NAV, which may affect the training. Also this dataset is quite small, with only 50.000 training samples, which usually is not sufficient for a decent translation model.

The subtitles dataset however, contains a larger amount of sentence pairs, but the sentences are usually shorter, due to being spoken lines. Since the data is user-submitted not all translations are of good quality, or misaligned (Lison and Tiedemann, 2016). For example, "Kom igjen, Carolyn, få høre." is given as a translation of "That’s what I’m working on".

The choice of using a book for validation is to evaluate the performance of our model in a shifted domain, as fictional literature is relatively different in its written language than both government websites and movie subtitles. Furthermore, the “Hound of the Baskervilles” book and its translation are old, so its language is more archaic than that of our training sets, which provides for further domain distinction.

An overview of the datasets, their sizes, average sentence lengths and token-type-ratio (TTR) can be seen in table 1. For our datasets, the TTR is defined by the number of unique words divided by the total amount of words in each dataset. This gives us a rough insight on the diversity and relative size of the datasets. The higher TTR, the more the dataset is composed of unique words, and lower TTR might also suggest a larger size for the dataset.

Finally, for experimentation we have created more specific subtitle datasets, which will be further discussed in section 4.

3.2 Tokenizer

Since we are working on a machine translation problem, the vocabulary for our tokenizer is shared with the language pair for our translation direction, English to Norwegian, as this helps generalization of our model. The tokenizer splits sentences into words, and subwords, with each segmentation getting an ID to use in an embedded

[2] http://data.europa.eu/88u/dataset/elrc_1061

[3] https://www.nb.no/sprakbanken/en/resource-catalogue/oai-clarino-uib-no-parallel-nob/

[4] https://opus.nlpl.eu/OpenSubtitles-v2018.php

[5] https://farkastranslations.com

[6] https://tatoeba.org/

Dataset	Train	Validation	Test	Avg. no length	no TTR	Avg. en length	en TTR
Government	50 000	2 500	-	13.85	9.66%	16.81	4.06%
Subtitles	250 000	2 500	-	6.16	8.72%	6.76	6.61%
Book	-	2 500	-	13.53	20.23%	13.83	18.90%
DIY	-	-	101	5.57	57.37%	5.65	60.42%

Table 1: Overview of datasets and their usages, with amounts of sentence pairs, average sentence lengths, and type-token ratio (TTR) for each language. If a dataset has both training and validation parts, the TTR is calcuated for the training subset.

layer that creates numeric vector representations of each token. Our tokenizer used BPE (Byte-Pair-Encoding), with BPE-dropout, which keeps the most frequent words intact while splitting the rare ones into multiple tokens, and regularizes the token-splitting by stochastically corrupting the segmentation procedure (Provilkov et al., 2020). This is proven to be a powerful regularization technique for machine translation.

3.3 Model

The architecture of our model is shown in figure 3. It is an implementation of the aforementioned transformers architecture.

For feeding the data into our model, instead of defining the batch size of sentences it should process at each iteration, we use a batch sampler to get a specified amount of target tokens in each batch. Each input has been padded where necessary to make the representations have equal shapes. The input to the model is a batch of source and target token IDs from the tokenizer, and their respective attention masks that define what token IDs to give attention to in the forward call.

First the source and target are embedded to create numerical vector representations of each token with a specified size for the hidden state. Then we use the transformer's encoder, which uses the mechanisms of multi-headed self-attention to capture the meaning of the sequence and save these weights. The decoder then decodes the target sequence with these weights before passing through a classification layer, which in turn gives the estimated token IDs of the output sequence. We can then decode these with our tokenizer to get the prediction sequence. To generate the final predicted sentence with our model, we use different search methods as mentioned, which are described further in the next section.

Our transformer model's encoder and decoder, with its attention mechanisms, implements several algorithms inspired by other publications, namely "pre-norm residual connections" by (Nguyen and Salazar, 2019), "position-infused attention" by (Press et al., 2021), and "GLU non-linearity" by (Shazeer, 2020).

We backpropagate and train our model by calculating the cross-entropy loss of the predicted tokens against our target tokens, and use the AdamW optimizer and a linear learning-rate scheduler to tune our model over the epochs of training.

3.4 Search & Evaluation

As mentioned, in this project we use two different search methods, greedy search and beam search, to construct the final predicted sentences of our model. We have used greedy search for development of our models, and beam-search for the final evaluation, because greedy search is faster for shorter training time, but beam search gives better final predictions. Search is performed by decoding the output iteratively, token by token, and searching for the most probable translation, which is finally outputted. The SacreBLEU score is then used to evaluate our models numerically, but we also read the translations and judge them ourselves, because, as stated, it is difficult to give a score if a translation is decent or not. Some of our final translations and their respective BLEU-scores is given in the results section. For validating our model during training we use the book dataset and we will report the BLEU-score of the best epoch during training. For final evaluation we use the DIY dataset and average the BLEU-score on the dataset after 10 runs. For the results in this paper, the BLEU-score is multiplied by a factor of 100, since BLEU is defined between 0 and 1, and we find this more readable.

4 Experiments

To evaluate the performance of our model, we trained several versions of the model with different datasets and hyperparameters, examining transla-

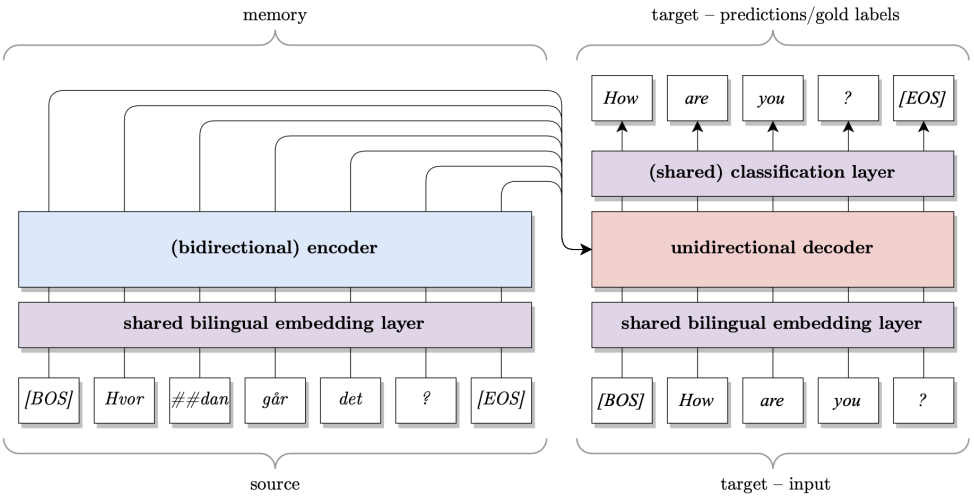


Figure 3: Architecture of our model. The encoder and the decoder are taken from a transformer model, which utilizes the attention mechanisms described in subsection 2.2.

tion quality as well as training time and file size of the final model. We used one of our computers, with an Nvidia RTX 3080 GPU and an Intel i5-12600K CPU, to train the models. Computation was done using the GPU's CUDA cores.

As a baseline for the tests, we chose the following hyperparameters:

- Number of encoder layers: 2
- Number of decoder layers: 2
- Hidden state size: 256
- Vocabulary size: 30 000 (tokens)

In additions to these parameters, the following parameters were used for training all models:

- Batch size: 1024 (target tokens)
- Dropout rate: 0.25 (25%)
- BPE dropout rate: 0.1 (10%)
- Number of attention heads: 16
- Learning rate: 10^{-3}
- Number of epochs: 100

All models were evaluated during training using the book dataset, and were tested using our own dataset.

4.1 Training sets

The first experiment was to determine which training set gives rise to the best translations. We trained one model on the government dataset, one model on the subtitle dataset, and one model on the combination of these datasets. Our hypothesis is that the combination would give the best translations, as this is the biggest dataset, and in general we are working with less data than in traditional machine learning tasks. Also the combination of the language used in government owned websites and the subtitles would provide a better generalization of translation functionality than training two models on separate data.

4.2 Hyperparameter testing

Once we found the dataset that resulted in the highest-scoring model, we trained several models on this dataset using different parameters. As our goal is to reduce the size and training time of our model, we chose to reduce the number of encoder and decoder layers, as well as reducing the hidden state size and vocabulary size. We tried several combinations of these factors and discuss this further in the results section.

4.3 Specific subtitle datasets

We noticed that the OpenSubtitles dataset was easily accessed through the Huggingface API and that in the meta-information for the subtitle entries in the dataset, the movie or TV-show the subtitle line was taken from was referenced by an IMDb-ID, IMDb being one of the largest freely accessible information repositories about movies and TV-shows.

By searching for the IMDb-IDs of specific movies we could filter datapoints in order to use subtitle data from specific movies of our choice.

This resulted in the creation of a Lord of the Rings (LOTR) dataset and a Harry Potter (HP) dataset, with subtitles from only the Lord of The Rings/The Hobbit and Harry Potter movies, respectively. These datasets were then combined into one dataset, the LOTR+HP dataset. This was done on the assumption that the manner of language used in these movie series is quite specific, and that training our model on this data would result in translations that mimic the way of speaking in the respective movies.

As the LOTR+HP dataset and the subtitle dataset are both subsets of the Huggingface OpenSubtitles dataset, we checked these datasets to see if there were any common datapoints between them. We found that in the training subset of the subtitle dataset, there were no sentences in common with LOTR+HP, but in the validation subset, there were eight sentences in common. As we did not use the validation subset in our experiments, we do not consider this to be a problem at all, but even if we did, the amount of overlap between the datasets is so small that it can be ignored.

In table 2, an overview of these created datasets can be seen, with their average sentence length for each language. Since the datasets for themselves are quite small, we decided to only use the combined dataset of the LOTR and HP movies, and use them only to train a specific model to analyze if the language of the translations is affected by the source material.

5 Results

When discussing the models in this section and in section 6, we refer to "smaller" models as not only the size of the model (its file-size, network-size, and trainable parameters), but also the amount of data it is trained on, which impacts how long it takes for the model to train fully.

See table 3 for an overview of the tested models and their scores. In general we can see that our models perform better on the final test set, than on the validation set during training. This is likely due to quite complex language in the domain shift of the validation set in addition to long sentences, and our test set having quite simple "every-day" language and short sentences. We will compare both when analyzing the models onward.

5.1 Training sets

As expected, we can see that our best performing model was trained on both the government and subtitles data combined, and achieved a final BLEU-score on the test set of 43.48, and also has the highest validation BLEU-score of 12.39. When prompting the model with some new sentences we came up with, we got some fairly decent translations like:

Src: There is one thing on my mind
Pred: Det er én ting på hjertet mitt
There is one thing on my heart[7]

What is interesting with this example is that the model seems to have understood a lot of the sentence, and not directly translated each word, as in English we would say we have something on our mind, but in Norwegian we would say we had something on our "heart", i.e. *"Å ha noe på hjertet"*, which it seems like it tries to convey, even though its translation is not perfect. It also has an acute on the e in "én", which suggests that the model might understand that it is “one” thing and not “a” thing, as these words are the same in Norwegian, but differentiated by the acute. Here are some more examples:

Src: Out of all these things that I’ve done, I think I love you better now.
Pred: Av alle disse tingene jeg har gjort, tror jeg at jeg elsker deg bedre nå.
Of all these things I have done, I think that I love you better now.

Src: I am looking forward to summer vacation, but I am going to miss IN5550
Pred: Jeg gleder meg til sommerferie, men jeg kommer til å savne INR50
I look forward to summer vacation, but I am going to miss INR50

As we can see the translations are quite satisfactory, it struggles to translate IN5550 as this is outside the vocabulary, but it still gives a reasonable code which fits in the sentence.

The model trained only on the government data performed worse, with only a test BLEU of 6.33 and a validation BLEU of 3.88. Trying to prompt it with one of the same sentences we get:

[7]The final italic sentence is our literal translation of the prediction back to English, for readability of those who do not know Norwegian. Here we had to assume the translations of some incorrectly spelled words.

Dataset	Train	Validation	Test	Avg. no length	no TTR	Avg. en length	en TTR
HP	11 105	-	-	5.88	17.13%	6.15	15.09%
LOTR	9 531	-	-	5.36	20.08%	5.95	16.54%
LOTR + HP	20 456	-	-	5.65	15.56%	6.06	13.03%

Table 2: Overview of specific datasets and their usages, with amounts of sentence pairs, average sentence lengths, and TTR per language. The datasets were searched and split manually, which is why the combined dataset has fewer data points than the sum of its constituents.

Dataset	Hidden size	Vocab. size	Layers
LOTR+HP	256	30 000	2
Government	256	30 000	2
Subtitles	256	30 000	2
Gov+Sub	256	30 000	2
small1	64	10 000	1
small2	64	30 000	1
small3	128	20 000	1

Table 3: The hyperparameters chosen for each model. The models small1, small2, and small3 are trained on Gov+Sub data.

Dataset	File size	Tr. speed	Tr. time/epoch	Params	Valid BLEU	Test BLEU
LOTR+HP	209 MiB	23 it/s	8.7 sec	18.4M	3.53	8.64
Government	209 MiB	32 it/s	33.6 sec	18.4M	3.88	6.33
Subtitles	209 MiB	26 it/s	1 min 45.0 sec	18.4M	11.75	39.64
Gov+Sub	209 MiB	25 it/s	2 min 34.4 sec	18.4M	12.39	43.48
small1	17 MiB	55 it/s	1 min 18.4 sec	1.42M	5.99	29.10
small2	47 MiB	37 it/s	1 min 44.3 sec	4.00M	6.70	31.84
small3	64 MiB	46 it/s	1 min 26.9 sec	5.57M	9.42	39.20

Table 4: The file sizes, estimated training speeds, estimated training time per epoch (not counting validation), and (mean) BLEU scores for each model trained using specified datasets and hyperparameters, as seen in table 3.

Src: I am looking forward to summer vacation, but I am going to miss IN5550
Pred: Jeg vil sende å se utsatt fersøk, men jeg vil vise til at familiemedlemmer
I will send to see delayed ettempt [sic], but I will refer to that family members

This translation is not satisfactory, but bears some resemblance to the original sentence. Also, the choice of translated words is interesting, like "sende", "utsatt fersøk", and "familiemedlemmer", which are probably common words in the government dataset. If we test this assumption further by giving the model a sentence similar to what we would find on a government website, we see that it performs much better, and also have learned some names:

Src: To apply for a grant you have to contact NAV
Pred: For å søke om tilskudd du må kontakte NAV[8]
To apply for a grant you must contact NAV

We also assumed that our combined LOTR+HP model would have the same traits as the government model, that it uses the specific "fantasy" glossary, and while it might not be the best generalized model, it is better suited for practical translations of fantasy related sentences. It performed slightly better than the government data with a test BLEU of 8.64 and a validation BLEU of 3.88. We can see that our assumptions is fairly correct when looking at the next two sentences, it still struggles somewhat on basic words and sentence structure, but has

[8]The word order in this prediction is ungrammatical in Norwegian. A better translation would be "For å søke om tilskudd *må du* kontakte NAV".

no trouble mentioning trolls, or translating names correctly:

Src: I am looking forward to summer vacation, but I am going to miss IN5550
Pred: Jeg ser ut hva som jeg skal undervistn, men jeg skal fessnangler troll med deg gode.
I look[9] *what as I will teachedn [sic], but I will "fessnangler" (???) trolls with you the good.*

Src: Neville Longbottom took all the Hobbits to Isengard
Pred: Nilus Langballe tkker Dumbled alle hobbiter til Jarnagard
Neville Longbottom thnks Dumbled [sic] all hobbits to Isengard[10]

The model trained on just the subtitle dataset performed quite well with test BLEU 39.64 and validation BLEU 11.75, only slightly worse than the combined model. This is likely due to the larger size of the dataset. There are no specific words or language use in this model, as the data is quite diverse. The translation is quite satisfactory:

Src: Out of all these things that I've done, I think I love you better now.
Pred: Av alle disse tingene jeg har gjort, tror jeg jeg elsker deg bedre nå.
Of all these things I have done, I think I love you better now.

Thus, we choose the dataset using baseline hyperparameters and the combined government+subtitle dataset as our baseline model.

For our smaller models trained on the combined dataset, performance decreases, as would be expected, but some of the translations are still quite good. This is a trade-off between smaller models taking less space, and having faster training and inference speeds, which can improve the overall practicality of the model.

Here is an example comparing outputs from models `small1`, `small2`, `small3`, and the baseline model:

Src: Out of all these things that I've done, I think I love you better now.
`small1`: Jeg har alltid gjort det jeg har gjort, jeg elsker deg.
I have always done what I have done, I love you.
`small2`: Av alle disse tingene som jeg har gjort, tror jeg elsker deg bedre nå.
Of all these things that I have done, I think love [sic] you better now.
`small3`: Av alle disse tingene jeg har gjort, tror jeg elsker deg bedre nå.
Of all these things I have done, I think love [sic] you better now.
Baseline: Av alle disse tingene jeg har gjort, tror jeg at jeg elsker deg bedre nå.
Of all these things I have done, I think that I love you better now.

5.2 Hyperparameter testing

As seen in table 3, it is apparent that the number of parameters, and thus the file size of the final model, is affected by the hidden state size and the number of encoder and decoder layers – the smaller the hidden state size, the better. The vocabulary size also has a significant effect on the file size as well as the number of parameters.

Vocabulary size had a major effect on the training speed. We observed a training speed of around 37 iterations per second (it/s) for `small2`, while `small3` trained at around 46 it/s and `small1` trained at around 55 it/s. However, these numbers do not account for the fact that most of the time training was spent on evaluation, making the total training time vary between three and four hours.

In addition to vocabulary size, the hidden state size, and numbers of layers affects the training speed as well, if one compares `small2` with the baseline model.

Another factor that affects training speed is the training dataset. The baseline model trains at 25 it/s, while the government dataset model trains at a faster 32 it/s, and the LOTR+HP dataset model trains at a slower 23 it/s.

The various hyperparameters have different effects on the BLEU score. Increasing the vocabulary size from 10 000 to 30 000 between `small1` and `small2` only increased the test BLEU score by 2.74, while increasing the hidden state size and numbers of layers to 256 and 2 between `small2` and the full-size baseline model increases the test

[9] as in "I look beautiful"
[10] The model correctly translates the names "Neville Longbottom" and "Isengard" to how these names are rendered in the Norwegian translations, "Nilus Langballe" and "Jarnagard". It also seems like it almost mentioned Dumbledore.

BLEU score by 11.64.

6 Discussion

Given the results in subsection 5.1, larger models seem to perform better than smaller models, which would be as expected. However, we also saw that for language-specific translations, like translating a message from the government, or a fictional fantasy description, the smaller models trained specifically on these types of data performed well. From this we could argue that it is more practical to train and inference smaller specific models, for the use-case that is wanted, rather than spending many resources trying to create larger generalized models.

An interesting detail to note about the dataset influence on training speed is that it is not affected by the amount of datapoints, as each iteration of the training uses a batch of datapoints, and not the full dataset. Instead, if one compares sentence length to training speed, it appears that the shorter the average sentence, the slower the training is: The LOTR+HP dataset model, despite its dataset having the fewest datapoints, has the slowest training speed of 23 it/s, with average sentence lengths of 5.65 and 6.06. On the other hand, the fastest big model was trained on the government dataset and had a training speed of 32 it/s with average sentence lengths of 13.85 and 16.81. This is most likely due to the batch sampler, which approximates the batch size to a specified amount of tokens, iterating through more samples when the average sentence length is short, to reach this amount. For this architecture, this suggests that the longer the sentences, the more feasible it is to train models quicker, but this likely varies based on how the batches are sampled.

As seen in section 5.2, we see that the smallest model that we trained, at 17 MiB, had a test BLEU score at 30, which indicates a fair quality of translations. However, we see that the bigger the models the better – the large models at around 200 MiB of file size had the best performance with little influence from choice of vocabulary size, and give better translations than the small models as the BLEU score goes above 40. This indicates that implementing transformer-based translation models in small-scale environments, such as smartphone apps and Internet of Things (IoT) devices, involves a compromise between resource efficiency and translation quality, but that small models can still be usable for translation from English to Norwegian.

On the other hand, vocabulary sizes noticeably affect the training time and file size, but given that they have a relatively small effect on translation quality, it may be beneficial to use smaller vocabulary sizes in order to train faster. However, since most of the training time is spent on validation, this still means that high-performance hardware is needed to train these models.

In general, depending on one's use case, we argue that smaller domain-specific models can be more practical than training larger generalized models. Examples of this could be a model for tabletop role-playing games, that translates fantasy descriptions for playing with a group where people do not know the same language, or a model that translates song lyrics or poems that tries to capture the meaning of the verses. These models could be trained on relatively small datasets, which can fairly easy be put together by a small web-scraping script or with user-submitted translation. Inference of the models would be fast and computationally easy enough to be performed locally on easily accessible devices, like smart phones and laptops.

7 Conclusion

In this paper we trained several transformer models for English-to-Norwegian machine translation with different sizes to compare the quality of the translations. We found that larger models tend to generalize better and have the highest quality of translations. However, we also found that the practicality of these models suffers from the need of large datasets and long training times, and on the scale of this paper, even our relatively "large" models struggle with domain-shift of translations. We propose that a better solution is to train smaller domain-specific models, to achieve the same quality of translations as larger models on each specific use-case. These smaller models also tend to learn their specific vocabulary better, and can translate specific words and names better. In general, we find it more practical, with regards to time and computations, to fit smaller domain-specific translation models to get the best quality translations for the least resources spent.

References

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).

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

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

Ofir Press, Noah A. Smith, and Mike Lewis. 2021. Shortformer: Better language modeling using shorter inputs. 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 5493–5505, Online. Association for Computational Linguistics.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Noam Shazeer. 2020. GLU variants improve transformer. *CoRR*, abs/2002.05202.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational linguistics*, 29(1):97–133.

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.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

French-English Translation of Artwork Description: A Neural Machine Translation Study

Nolwenn Marie Emilie Bernard
University of Stavanger
nolwenn.m.bernard@uis.no

Abstract

This paper explores the task of French to English neural machine translation. In particular, we are interested on the generalization ability of models trained with governmental and subtitles corpora to the domain of art. Indeed, providing a translation of artworks' description in French museum can improve the inclusion of international visitors. While machine translation has seen significant advancements, especially with the deep learning approaches, no specific model has been trained for our use case, and no specific dataset is available for training such model. Therefore, we investigate how well a model trained on unrelated limited resources can generalize to the art domain and others. We conduct experiments using different training datasets and model configurations to find the best model. For the evaluation, we use the well-known BLEU score. Our analysis reveals that it is possible to find a trade-off between the model size, complexity, and inference time. Ultimately, our goal is to experiment and start the discussion on model generalization when training data is limited.

1 Introduction

There are more than a thousand museums in France that are visited by domestics or internationals. Thus, a translation of artworks' description to English can be displayed for a better inclusion of international visitors. The translation of all descriptions would be tedious and slow task, therefore, we propose to develop a neural model to perform automatic translation of artwork description.

Machine translation is a well studied task in the domain of Natural Language Processing (NLP). Significant advancements have been done in the last few years especially thanks to the Transformer model (Vaswani et al., 2017) and more generally to large language models such as mT5 (Xue et al., 2021) and BART (Lewis et al., 2020). However, to the best of our knowledge no specific model has been specifically trained for the translation of artwork description and no dataset is available for this task. As the creation of such dataset, that is big enough for training, is time-comsuming, we study how well a model trained on limited resources unrelated to the art domain generalizes to it and to others.

In this paper, we perform experiments with different training datasets and model configuration, then we compute the BLEU score (Papineni et al., 2002) to evaluate the quality of the different models. After analysis of the results, we observe that it is possible to have a relatively small model that has similar results than a 4 times bigger one. For our use case, we select a Transformer-based sequence-to-senquence model that achieves on of the highest BLEU score on our artwork description dataset.

The rest of the paper is structured as follows. Section 2 provides background on neural machine translation and on existing approaches tackling this task. Section 3 presents the task in more details. In Section 4, we discuss the different datasets using for training and evaluation of the models. The different experiments performed are introduced in Section 5. Then, we analyse the results in Section 6. Finally, Section 7 concludes this work. All resources developed as part of this work are made available at: https://github.uio.no/nmbernar/nmt-fr-en.

2 Background

Deep learning techniques have significantly changed the field of machine translation. Indeed, deep learning models have exhibited considerable improvement of the translation quality compared to previous approaches. One of the primary architectures for machine translation is the sequence-to-sequence (Seq2Seq) model (Sutskever et al., 2014). A Seq2Seq model consist of an encoder network that represents the input sequence in a vector space with a fixed length, a decoder network that decodes

the encoded input to the target sequence word by word. This framework enables end-to-end learning, allowing the model to capture complex dependencies between the input and output sequences.

Later, the attention mechanism has been applied by Bahdanau et al. (2016) to further improve Seq2Seq translation models. The idea is to enable the decoder to focus on different parts of the input sequence by giving them different levels of importance, i.e., attention. This tackles the issue of Seq2Seq models without attention that need to encode all important information in a fixed-size vector.

Nowadays, Transformer-based models represent the majority, if not all, of the best performing models. The Transformer architecture was introduced by Vaswani et al. (2017) and revolutionized the field of NLP including machine translation by establishing state-of-the-art performances. This architecture is especially good at modeling languages thanks to its ability to capture long range dependencies.

Several evaluation metrics have been proposed to assess the quality of translation models (Chatzikoumi, 2020). The Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002) is the most commonly reported metric; it measures how close is the machine translation to a human translation that acts as the reference. However, it also has some limitations like its dependency on the tokenization technique and its difficulty to capture the fluency of the text. The metric for evaluation of translation with explicit ordering (METEOR) (Banerjee and Lavie, 2005) was proposed to tackle some of BLEU's limits by incorporating recall and linguistic features such as stemming and synonymy.

3 Methodology

In this section, we present in details the problem of machine translation (Section 3.1). Then, we discuss the tokenizer used in this work (Section 3.2) and the evaluation method (Section 3.3).

3.1 Problem description

This paper focuses on the translation of French sentences to English using a Seq2Seq neural network. Our Seq2Seq network f is composed of: an encoder network E, a decoder network D, and a classification layer CL with a size equal to the number of token in the tokenizer's vocabulary (Figure 1). The goal of f is to generate a target sentence $T = (t_1, ..., t_n)$ given an input sentence $S = (s_1, ..., s_m)$, where m and n are the length of the input and target sentences with regards to the number of tokens. The sentences S and T are constituted of tokens belonging to the tokenizer vocabulary. The generation of the target sequence is typically performed in an iterative manner.

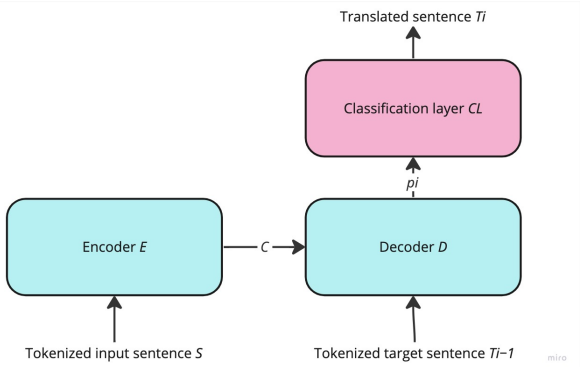


Figure 1: Simplified representation of the Seq2Seq model f

First, the tokenized sentence S is encoded in vector C, also referred as context vector, by feeding S to E, i.e., $C = E(S)$. We note here that C has a fixed-size if there is not an attention mechanism. Then, the decoder D can generate the translated sentence $\hat{T}$ based on C and previously generated token. The next token $\hat{t}_i$ is predicted by feeding C and $\hat{T}_{i-1} = (\hat{t_1}, ..., \hat{t}_{i-1})$ to the decoder, the output of the decoder is then sent through the classification layer, i.e., $p_i = CL(D(C, \hat{T}_{i-1}))$. We note here that $\hat{t_1} = [BOS]$, where $[BOS]$ is a special token representing the beginning of sentence. During training time of RNN-based f, we use threshold to decide whether $\hat{t}_{i-1}$ takes the ground truth token or not, i.e., the teacher forcing method (Goodfellow et al., 2016). Teacher forcing has the advantage to reduce convergence time. During inference, we apply a greedy decode method to the output of the classification layer CL to determine $\hat{t}_i$, this method always selects the most probable token, i.e., $\arg\max p_i$. The inference continues until the special end of sentence token is predicted.

3.2 Tokenizer

In this work, we train one custom subword tokenizer with a vocabulary of 20,000 tokens on French and English sentences in the Combined dataset (i.e., combination of Europarl and OpenSubtitles datasets). The tokenizer is trained with byte-pair encoding (BPE) algorithm (Sennrich

et al., 2016), based on their frequency subwords (also referred as tokens) are kept or not in the final vocabulary. BPE creates the tokens by iteratively merging the most frequently occurring pairs of consecutive bytes or characters.

3.3 Evaluation

The quality assessment of a translation is not a simple task, indeed, the richness and subjectivity of languages can lead to many different good translations for a single sentence. Consequently, in this problem of machine translation we combine a human evaluation by looking at cherry picked sentence pairs and an automatic evaluation using BLEU score (Papineni et al., 2002). We note here that using human evaluation is not practical in cases of much larger use cases, i.e., does not scale well. The automatic evaluation with BLEU allows for reproducible and objective quality assessment of the models as opposed to humans that can have divergent opinions. However, BLEU presents some limitations such as its dependency on the tokenizer that need to be taken into consideration during the analysis of the results.

The BLEU score is a well established metric to evaluate translation models and is known to correlate quite well with human evaluation. This score varies between 0 and 100 where a score of 100 indicates a perfect model, i.e., the translated sentences are identical to the references Papineni et al. (2002). We note here that a reference score indicating a good and understandable translation depends on parameters such as the language pair studied, the domain, and the number of reference translation.

4 Datasets

As stated in Section 1, a parallel corpus French-English with artwork description is not available at the time of the experiments. Therefore, we utilize four distinct French-English bilingual parallel corpora: (1) Europarl (Koehn, 2005), (2) OpenSubtitles (Lison and Tiedemann, 2016), (3) Book (Farkas), and (4) DIY. The first two datasets are used for training, while the last two server as test dataset to evaluate how well the models generalize. More details on each dataset is given in the Sections 4.1–4.4 and a summary is provided in Table 1.

4.1 Europarl

Europarl is a parallel corpus for different European languages, in this work we select a random sample of 50,000 French-English sentence pairs. This corpus comprises professionally translated texts from the proceedings of the European Parliament, this includes speeches, debates, and discussions around politics and legislation.

As the dataset contains texts from the European Parliament, the language used is rather formal and can include very specific words, e.g. noria and plénière, that are not common in the day to day language. We also note that the majority of the sentences are statements, as for example: "*La commission des affaires juridiques a émis un vote unanime sur ce rapport et les diverses positions et déclarations communes y annexées.*" translated to "*We had a unanimous vote in the Committee on Legal Affairs on this report and on the various common understandings and statements that were annexed to it.*" Therefore, it matches the type of sentences we use to describe pieces of art.

4.2 OpenSubtitles

This dataset contains unofficial subtitles for movies and TV shows, we note here that theses subtitles are produced by internet users that most likely are not professional translators and aligned based on the associated timestamps. Therefore, we expect the dataset to contains translations of variable quality due for example to intentional or not mistakes, timestamps mismatch, or how the original sentence was interpreted by the user performing the translation. The following sentence pair is an example of mistake: "*Quand même,je ne pensé pas-*" translated to "*Just the same, **I-I** don't think–.*" To respect the idea of having small dataset to create our model, we select a random sample of 252,500 sentence pairs from the original dataset.

After a closer look to the corpus, we make the following observations. First, the language used is mostly informal and modern. Second, the sentences are on average shorter and are usually part of dialogues which is consistent with movies or TV shows dialogues. Third, some special characters like ♪ introduce some noise in the dataset and may induce some biases in the models. Finally, this dataset includes diversity thanks to the different types of movies and TV shows available, hence, we expect the models to generalize better to other domains.

Dataset	Train	Validation	Test	Avg. len. fr sentences	Avg. len. en sentences
Europarl	50,000	2,500		18.5	17.8
OpenSubtitles	250,000	2,500		6.3	6.9
Combined (Europarl + OpenSubtitles)	300,000	5,000		8.4	8.8
Book		2,500		13.8	14.2
DIY			100	16.5	14.1

Table 1: Datasets details. The average length of sentence uses word as unit.

4.3 Book

This dataset is a translated and aligned version of the book "Hound of the Baskervilles" written by Arthur Conan Doyle in 1902. As mentioned before, this dataset is used as a test corpus because it belongs to a different domain than Europarl and OpenSubtitles.

Despite sharing the particularity of representing fictional corpus with OpenSubtitles, this dataset has specific characteristics with regards to its quality and the writing style. Indeed, the translator of the book has previous professional experience, therefore, the quality of the translation is expected to be better than for OpenSubtitles. In terms of writing style, the book has narrative sections some of which can be written with a first-person narrator. For example, "*Je me levai, ouvris ma porte, inspectai les alentours.*" is translated as "*I rose, opened my door, and peeped out.*" On the contrary movies and TV shows are in majority composed of dialogues with multiple characters. Moreover, languages can evolve over time, i.e., some expressions, idioms, and turn of phrase might disappear, hence this dataset can present such expression. Take the following pair as an example: "*Verriez-vous un inconvénient à ce que je promène mon doigt le long de vos bosses pariétales ?*"" translated to "*Would you have any objection to my running my finger along your parietal fissure?*", here "*bosses pariétales*" is an uncommon expression.

4.4 DIY

For this work, we need a French-English parallel corpus with sentences describing artworks. Since such a corpus is not available to our knowledge, we create our own with a total of 100 sentences. For the creation, we use ChatGPT (OpenAI, 2022), a large language model released late 2022. The sentences are generated using the following main prompt: "create 100 sentence pairs translating work of art description from French to English ", we note here that the prompt was slightly modified to including sentences about pop art, street, and modern art (see Appendix A). The sentences are manually checked by the author to ensure their accuracy and alignment.

Using a large language model like ChatGPT allows us to produce many sentence pairs that includes diverse artworks in a short amount a time. However, it is noteworthy that such models can produce grammatically incorrect text or be incorrect (Kasneci et al., 2023). For example, the pair "*La Persistance de la mémoire de Magritte est une interprétation surréaliste des montres molles.*" to "*Magritte's The Persistence of Memory is a surrealistic interpretation of melting watches.*" needs to be corrected because "The Persistence of Memory" is painting by Salvador Dalí. Moreover, during the generation, we observe that some sentence pairs are duplicated. That is the reason why a manual verification of the sentences is necessary. In total, 111 sentence pairs were generated among which 11 were removed and 4 were modified (e.g., correction of the artwork's title, artist's name).

The art domain is interesting because the description of a piece of art is likely to contain declarative statement using modern language, however it presents at least two challenges in our opinion. First, some artworks do not have a translated title, e.g., "Pietà" by Michelangelo, that adds a difficulty for the model as the translated sentence will contain both languages. Second, it is possible that a translated title is literally or semantically different from the original name such as "La Joconde" for "The Mona Lisa."

5 Experimental Setup

In this section, we describe the different experiments performed (Sections 5.2–5.3), the different model configurations used are presented in Table 2. After, we provide implementation details in Section 5.4.

Name	Backbone	# Heads	# Enc. layers	# Dec. layers	Emb. size	Hidden size	# Parameters
T2_512_512	Transformer	2	3	3	512	512	43,364,896
T2_512_256	Transformer	2	3	3	512	256	41,790,496
T2_256_256	Transformer	2	3	3	256	256	18,546,720
T4_512_256	Transformer	4	3	3	512	256	41,790,496
RNN_3	RNNs		3	3	256	256	13,418,016

Table 2: Models configuration.

5.1 Backbone Architecture

In this experiment, we try two backbone architectures for the sequence-to-sequence model. More specifically, we train the model RNN_3 with recurrent neural networks (RNNs) and the model T2_256_256 with a Transformer. The goal of this experiment is to observe the difference in term of performances between Transformer that is becoming the new "norm" and RNNs that are loosing attention for neural machine translation.

5.2 Hyperparameters Variation

The choice of hyperparameters can have a significant influence over the performance of a model. In this study, we only perform experiments on three hyperparameters: (1) number of attention heads, (2) size of the embedding layer, and (3) hidden size. For these experiments, we train the models with the Combined dataset.

The attention head plays a role in the capture of relevant dependencies. Therefore, we study if a larger number of attention heads, that should help the model to find more dependencies, provides better translation. We try two sizes for the embedding layer: 256 and 512. The embedding layer creates the word representation of the input and target sentence, a large size allows the model to capture more information potentially leading to a better representation of the sentences. However, by increasing the size of this layer, the number of parameters in the model also increases; this can negatively impact the inference time that is an important property in our study. Finally, we vary the hidden size to study the trade-off between expressiveness of the model and translation quality.

The influence of other hyperparameters such as the number of layers in the encoder or decoder is left for future work.

5.3 Training Dataset

For this study, we have two training datasets in different domains (i.e., government and subtitles). In addition to these, we create a new one Combined that include Europarl and OpenSubtitles. The idea is to leverage the specificity of each dataset and provide diverse examples for the model to learn from. We think that a diverse set of example could beneficit the generalization ability of the models. To verify this hypothesis, we train models with one of these dataset and evaluate them on the Book and DIY datasets.

5.4 Implementation

The different models are implemented using the machine learning framework PyTorch.[1] The evaluation of the models is performed using the SacreBLEU (Post, 2018) implementation of the library HuggingFace Evaluate.[2]

In terms of training, each models is trained for 25 epochs with early stopping if the loss on the validation dataset does not improve for 3 epochs in a row. The batch size is set to 128 and we use the Adam optimizer (Kingma and Ba, 2017) with β_1, β_2, and ϵ to 0.9, 0.98, and 1e-9 respectively. The training is performed on either a Tesla P100 or Tesla V100 GPU.

6 Analysis

In this section, we analyze the performances of the different models trained with regards to the experiments described in the previous section. Additionally, we examine cherry-picked examples produced with our best model. Table 3 presents the results of the experiments performed.

First, we look at the BLEU scores obtained by the model RNN_3. We observe that the scores are almost 0 denoting very poor translations, i.e., very

[1] https://pytorch.org

[2] https://huggingface.co/docs/evaluate/index

Model	Training dataset	BLEU validation	BLEU Book	BLEU DIY	Avg. inference time on DIY (s)
T2_512_512	Europarl	25.82	7.02	7.99	0.41
T2_512_512	OpenSubtitles	**39.60**	10	9.60	0.35
T2_512_512	Combined	24.19	**13.64**	**14.56**	0.36
T2_512_256	Combined	24.54	**13.64**	14.31	0.34
T4_512_256	Combined	25.28	13.50	13.18	0.35
T2_256_256	Combined	24.89	13.38	14.45	0.21
RNN_3	Europarl	0.04	0.03	0.06	**0.04**

Table 3: Evaluation of the models on the associated validation, book and DIY datasets. The average inference time was computed using CPU.

far from human translations. We note here, that the training of the model was successful and we observe a gradual decrease of the loss on both the training and validation set until early stopping at the epoch 16. The examination of the predictions on Book and DYI shows that the predictions are not grammatically or semantically correct. The examples below illustrate this:

1. **Source**: “La Joconde est un portrait célèbre peint par Léonard de Vinci.""
   Target: “The Mona Lisa is a famous portrait painted by Leonardo da Vinci.""
   Prediction: “„ the the the the the the the the the the the the the""

2. **Source**: “– Les stores ne sont pas baissés.""
   Target: “"The blinds are up.""
   Prediction: “is the the the the the the the.""

It appears that the model RNN_3 is only able to predict very frequent words or punctuation and does not capture the sentence meaning. We hypothesis that the model is not complex enough to create word representation with sufficient information or to detect non-trivial dependencies. More experiments shoud be performed in future work to confirm or not our hypothesis. The rest of the models that uses Transformer considerably outperform RNN_3 with BLEU score of 24-25 on validation dataset.

Now, we focus on the analysis of the influence of selected hyperparameters: number of attention heads, the size of the embedding layer, and the hidden size. We look at the models T2_512_512, T2_512_256, T2_256_256, and T4_512_256 trained with Combined in Table 3.

Hidden size. In terms of BLEU scores, T2_512_512 and T2_512_256 achieve the highest scores for the Book dataset (13.64) and the DIY dataset (respectively 14.56 and 14.31). This shows that reducing the hidden size has a minimal impact on the generalization ability of the model while slightly reducing the size of the model as well as the average inference time.

Number of attention heads. When comparing the models T2_512_256 and T4_512_256, we see that T4_512_256 performs better during the validation phase with 25.28 against 24.54 for T2_512_256. However, same observation is not true when testing with Book and DIY. We suppose that the additional dependencies found by T4_512_256 compared to T2_512_256 are more domain specific, hence, penalizing the model when performing translation on unknown domains.

Embedding layer size. We study the model T2_512_256 and T2_256_256 that have embedding layers with different sizes. The first thing to highlight is the significant difference between models' size, a direct illustration of this is the slower average inference time for T2_256_256. Moreover, despite being slower, T2_256_256 performs better on the validation and DIY datasets, nonetheless, the difference is small. Similarly to hidden size variation, this experiment shows that it is possible to keep similar quality of translation while reducing the size of the model and consequently the average inference time.

Finally, we examine the results of the models T2_512_512 trained on the three different datasets. The model trained OpenSubtitles achieve the best BLEU score on its validation set, however, we cannot compare this score with the other models as the validation is model specific. Therefore, we focus our analysis on the scores obtained on Book and DIY. We observe that the models trained with

Europarl and OpenSubtitles struggle to generalize to unknown domains as illustrated by significant drop with regards to BLEU scores. The model trained with Europarl achieve the worst performance, this might be explain by the characteristics of the dataset (e.g., linguistic style and vocabulary used) that do not necessarily match with the ones of Book and DIY. It is noteworthy that the idea of combining OpenSubtitles and Europarl to create a bigger and more diverse training dataset allowed us to reach the best performances among all the models, thus, improving the generalization ability of the model.

After performing the different experiments, we find that a trade-off is possible between model size, complexity, and translation quality. Indeed, the experiments with different hidden and embedding layer sizes show that a smaller model can achieve almost the same quality of translation with reducing the resources necessary to store and the inference time. Taking this into consideration, we believe that the model T2_256_256 trained with Combined is the best for our application of artwork description translation. We examine some translation examples produced by T2_256_256 on DIY in Table 4. We observe that the model can struggle to translate artwork's title and artist's name that are named entities, this is consistent with the expected challenges presented in Section 4.4.

7 Conclusion

In this work, we study the problem of neural machine translate from French to English. More specifically, we study how well models generalize to a new domain when trained on relatively small datasets.

We perform experiments playing with the hyperparameters, the training dataset, or backbone architecture of the translation model. We find out that a trade-off between model size, complexity, quality is possible. Our best model is the model T2_256_256 trained with Combined, it achieves the BLEU score of 14.45, and has an average inference time of 0.21 seconds on DIY.

More experiments with the hyperparameters of the models could be done in future work, e.g., vary number of encoder/decoder layers or the activation function. In addition to a more extensive exploration of models with RNNs as backbone architecture.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

Eirini Chatzikoumi. 2020. How to evaluate machine translation: A review of automated and human metrics. *Natural Language Engineering*, 26(2):137–161.

Andras Farkas. Bilingual books. Accessed: 2023-04-21.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, Stepha Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103(102274).

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X: Papers*, pages 79–86.

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

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*, pages 923–929.

OpenAI. 2022. Introducing chatgpt. Accessed: 2023-05-11.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, ACL '2002, pages 311–318.

Source sentence (fr)	Predicted translation (en)	Target translations (en)	Comments
La Joconde est un portrait célèbre peint par Léonard de Vinci.	The Jocirium is a famous portrait by Leonard of Vinci.	The Mona Lisa is a famous portrait painted by Leonardo da Vinci.	The Jocirium is not an English word, here we see that the translation of title is challenging. The translation of the artist's name is also incorrect.
Les Noces de Cana de Véronèse est une peinture biblique qui montre un festin de mariage.	The Noces of Cana of Veroes is a biblical painting that shows a feast of marriage.	Veronese's The Wedding Feast at Cana is a biblical painting depicting a wedding feast.	Same as the previous example, we observe an error in the artwork's title translation while the rest of sentence is consistent with the target.
La Nuit étoilée de Van Gogh est une représentation expressive du ciel étoilé et du village.	Van Gogh is a show expressive performance of the beginning of the beginning of the beginning and the village.	Van Gogh's The Starry Night is an expressive representation of the starry sky and the village.	This example highlights the grammatical struggle of the model, the sentence does not make sense and we see a repetition of the 3-gram.
Les Iris de Van Gogh sont une représentation vivante et colorée de fleurs d'iris.	Van Gogh are a alive performance and colored flowers.	Van Gogh's Irises are a vibrant and colorful representation of iris flowers.	Here we see that the translation provided by the model is shorter than the target, thus, some key elements are missing.
La toile de Robert Indiana avec le mot LOVE en lettres capitales est devenue un symbole universel de l'amour et de l'optimisme.	Roberta's paintinga with the word LOVE word in letters became a universal symbol of love and optimism.	Robert Indiana's canvas with the word LOVE in capital letters has become a universal symbol of love and optimism.	This example illustrates the difficulty to translate the artist's name.

Table 4: Example translations of sentence pairs in DIY by T2_256_256. Text in red highlights translation errors.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, WMT '18, pages 186–191.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL '16, pages 1715–1725.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, NIPS '14, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, NIPS '17, pages 5998–6008.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL '21, pages 483–498.

A Prompts for DIY

The prompts below are given to ChatGPT subsequently to create the DIY datasets:

1. create 100 sentence pairs translating work of art description from French to English
2. continue generation
3. more examples with street art
4. more examples with pop art
5. more examples with modern art
6. more examples with Norwegian artist

We note here that the prompt 2 was used several times in order to overcome the generated text limit with regards to the number of tokens. On average 15 sentence pairs are generated.

Guessing Menu Ingredients with a Transformer: A Qualitative Analysis of the Training Corpus of a Norwegian-to-English Transformer Model Used for Menu Translations

Amir Basic, Cornelius Bencsik and Torstein Forseth

Abstract

This paper looks at how a state-of-the-art model, the Transformer, performs when translating restaurant menu data from Norwegian to English when the it's trained on data from different domains, such as movie subtitles and sentences scraped from government websites. Our main analysis paid extra attention to the datasets in order to get a better understanding of the training corpus and a seperate corpus "a tourist visiting a restaurant in Norway" as the hypothetical scenario. The results showed that the model is prone to overfitting on the domain it was trained on. The model predictions were especially bad when it had to translate a word it never had seen before.

1 Introduction

Machine translation using large language models has experienced a fundamental change with the introduction of transformer-based models, which have transformed the way we approach natural language processing (NLP) tasks. The transformer architecture has become the standard for building large language models for NLP. The transformer model (Vaswani et al., 2017) replaces the traditional recurrent neural network-based architectures with self-attention mechanisms, which enable the model to selectively attend to different parts of the input sequence, leading to more efficient information flow. This led to the introduction of the BERT model (Devlin et al., 2019), which is a pre-trained transformer model that is still used for fine-tuning to a wide range of NLP tasks. The transformer architecture has been widely adopted in popular translation systems such as Google Translate. Additionally, the success of large transformer-based language models like GPT-3/4 has shown the potential of pre-trained models for machine translation tasks, with Chat-GPT being the current go-to tool for most text-based tasks.

In this paper we investigated how the transformer model performs on when the model is tested on an out of domain dataset. We performed an in-depth data analysis on four data sources from different domains, with a focus on how the variation in the training corpuses affects the performance of the model. The Sennrich paper (Sennrich et al., 2016) showed that the transformer model performs well in low-resource settings, even with limited parallel data. We analyzed if that is the case in our setting with limited data, but with variation in type of words across the datasets. Quantitatively we measured performance using bilingual evaluation understudy (BLEU) score and qualitatively we manually inspected sentences to see how the decoded sentences differs the gold label sentence. The prior hypothesis is that training the model on a combination of the datasets should yield better performance than training a model for each of the datasets isolated as the latter models wouldn't capture the variation across the datasets. We have also chosen a practical setting by predicting on a dataset we have made ourselves based on restaurant menus. The hypothetical scenario is tourists who visit Norwegian villages could translate their food and drink menus. Given how specific the data domain of restaurant menus are we expected that this would pose performance challenges, especially due to the potential of out-of-vocabulary words.

2 Background

Transformers have demonstrated superior quality in machine translation tasks as stated by the original transformer paper (Vaswani et al., 2017). As also stated by the original paper, the amount of data needed for the model to perform well is significant. The experiments conducted in the original paper uses 4.5 million sentence pairs which is significantly more than what we trained our model on. (Jiang et al., 2020) has looked into how we can obtain information from the training corpus to help us understand what the transformer will predict on out of domain data. This work inspired us to look into

our training corpus to get a better insight on how it performed when translating Norwegian menu texts.

3 Methods, Algorithms and Metrics

3.1 BPE Tokenization

Byte-pair encoding algorithm (Sennrich et al., 2016) compresses the data by replacing the most common pair of sequential bytes of data with a byte that is not present in that data. BPE tokenization ensures that the most frequent words are represented as a single token and not split up, while less frequent words are split into subword tokens. The BPE algorithm allows us to use the powerful regularization method BPE dropout (Provilkov et al., 2020).

3.2 Search

The decoding algorithm searches for the most probable translation by iteratively decoding the output token by token. We use the greedy search, a fast method which greedily constructs the output by taking the most probable next-token predictions. The search algorithm relies on the model to provide the next-token predictions.

3.3 BLEU

Language generation models and machine translation models are difficult to objectively measure since there is not a single correct translation, due to sequence-to-sequence prediction and vocabulary complexity. Evaluation of the model is achieved through the bilingual evaluation understudy (BLEU) score (Papineni et al., 2002), which is comparable to a human interpretation of correctness. The BLEU score measures the similarity of the predicted translation to the given translation by taking the average of the precision of n-grams between the two translations, and adding a penalty based on translation length. This makes the BLEU score able to capture similarity between two translations without being completely identical at a word-level. However, the BLEU score is not perfect as explained by (Post, 2018), therefore, data analysis was a major part of the evaluation process, as well as our own judgment of meaning preservation, degree of fluent language translation, and possible linguistic phenomena problematic.

4 Data

Five data corpuses from four sources are used for training, validation, evaluation, and testing. All data corpuses consist of a Norwegian sentence paired with the English translation. Three of the corpuses are mainly used to train models, the government corpus, the Subtitles corpus, and the combined corpus which is a combination of the government and the subtitles corpus. Additionally, we have two corpuses that are utilized for validation and testing, the Book corpus and our custom made (DIY) corpus.

4.1 The Government Corpus

The government corpus is created from the Public Bokmål-English Parallel Corpus (PubBEPC), and the Bilingual English-Norwegian Parallel corpus from the Office of the Auditor General (Riksrevisjon) website. The data is constructed from publicly available Norwegian-English sentence pairs from www.riksrevisjonen.no, www.nav.no, www.nyinorge.no and www.skatteetaten.no. The data is expected to be translated from professional translators, and is from government websites, which indicate high quality in terms of translation and language. The corpus is expected to have complex language structure, an academic vocabulary, and a specific language.

4.2 The Subtitles Corpus

The subtitles corpus consists of Norwegian-English unofficial subtitles sentence pairs published by internet users to www.opensubtitles.org, that have been created into a large parallel corpus (Lison and Tiedemann, 2016). The corpus is expected to have more daily speech, complemented by simple language and vocabulary.

4.3 The Combined Corpus

A combination of the government and the subtitles data, containing all data from both corpuses. The combined corpus is an attempt to balance the complex and day-to-day language and specific domains from both datasets.

4.4 The Book Corpus

The book corpus is the translated and aligned book "Hound of Baskervilles" by Arthur Conan Doyle. The book is a part of the collection "Bilingual book" by FarkasTranslations.com. The corpus domain is slightly different from training data, which makes it great for evaluation.

4.5 The DIY Corpus

The DIY dataset consists of sentences from restaurant webpages. The sentences are a mix of information, dish descriptions, ingredients, and other aspects of the restaurant webpage. The dataset consists of 102 sentences from 13 different restaurants in Oslo. All restaurants had a Norwegian and English version of their webpage and menu. The choice behind the DIY dataset is the applicable use case for translation (tourists eating at a restaurant without an English menu). Additionally, there is also a mix of words that seems like a complex test for our translation models. The domain of the DIY corpus is different from the training corpus, and contains a restaurant-specific vocabulary and sentence structure, which includes a lot of listing of ingredients.

4.6 Data analysis - Norwegian

The corpuses varies in size, content, vocabulary, and complexity. The differences are a result of their sources and their domain, and have a possibility of affecting the outcome. Therefore, to be able to understand the difference in outcome and performance, it is crucial to understand the differences in the data. The size of the training corpus for the government dataset and the subtitles dataset are quite different, with the government dataset only having 50 000 sentence pairs and the subtitles dataset having 250 000 sentence pairs. The combined data is a combination of the two, and hence the data analysis part will mostly focus on the differences between the government and subtitles data (table 2).

For input data (the Norwegian sentences) the structure of the sentences varies. The government data has as expected a more complex sentence structure and language, with over twice as many words per sentence on average, and longer words.

Norwegian data analysis - structure				
Data	**sentences**	**word/ sent**	**char/ sent**	**char/ word**
Government	50 000	13.85	94.24	6.81
Subtitles	250 000	6.16	31.62	5.13
Combined	300 000	7.44	42.06	5.41
Book	2 500	13.52	72.74	5.38
DIY	102	12.22	77.92	6.38

Table 2: Data statistics from the Norwegian part of the data focusing on sentence structure and language complexity,

Another important aspect is the vocabulary of the training data. Even though the subtitles data has more sentences than the government data by a factor of five, it has less than twice the vocabulary size. The two vocabularies only share 10 569 out of 38 270 and 71 774 words. The subtitles data have a greater rate of words with appearance frequency less than or equal to one and five, which suggest a less complex language. Supporting this, the percentage of the total subtitles corpus that are the top 10, 100, and 1000 most frequent words from the corpus are higher than the government data, even though it has more words and a bigger vocabulary. Words in the top 100 most frequent subtitles words that are not in the top 100 government most frequent words are typically daily speech words and informal pronouns not fit for government documents: "jeg", "vi", "han", "meg", "så", "bare", "nei", "kanskje", "aldri".

On the other side, words in the top 100 government most frequent words that are not in the top 100 subtitles most frequent words are more formal, academic and government specific language such as: "Norge", "gjelder", "grad", "departementet", "prosent", "undersøkelsen", "tiltak", "dersom".

Norwegian data analysis - vocabulary						
Data	**Vocab size**	**word freq <= 1**	**word freq <= 5**	**top 10 words**	**top 100 words**	**top 1000 words**
Government	38 270	48.25%	77.85%	22.6%	46.4%	71.1%
Subtitles	71 774	57.10%	83.42%	23.1%	57.7%	79.4%
Combined	99 475	54.48%	81.55%	21.26%	52.84%	73.78%
					Vocab shared /w combined	**Shared vocab% of total data**
Book	4 627	55.46%	85.61%	19.73%	80%	94.52%
DIY	667	76.28%	96.78%	21.57%	76.43%	84.58%

Table 1: Data statistics from the Norwegian part of the data focusing on language and vocabulary.

Additionally, the government data has a lot of specific words that appears at least 300 times, which might not be very helpful for predictions in other domains, such as: "departementet", "nav", "riksrevisjon", "utenriksdepartementet", "eøs".

Examples of sentences from the government corpus: "Utgifter til losji som overstiger 10 000 kroner per år er kun fradragsberettiget når dette er betalt via bank eller ved trekk i lønn." and "Dokumentasjon vedrørende REDD+ under FNs klimakonvensjon". Examples of sentences from the subtitles corpus: "Jeg gjorde ikke noen ting." and "bli der dere er". The validation data has fewer sentence pairs and hence smaller vocabularies. The most important aspect of the validation set analysis is verifying that the words in the validation data are in the training data, which verify that the model has seen the Norwegian words. The book corpus shares 80% of its vocabulary with the combined vocabulary, which account for 94.52% of the total words in the whole book corpus. The DIY data share 76.43% of its vocabulary with the combined vocabulary, which account for 84.58% of the total words in the whole DIY corpus (table 1).

4.7 Data analysis - English

The English part of the data is a reflection of the Norwegian part, especially when it comes to sentence structure and complexity. Longer Norwegian sentences are usually translated to longer English sentences. This is true for the sentence structure and complexity for all data sets. Therefore, for the English part, vocabulary analysis is the most important part. For both the government and subtitles data, the total number of words increases, while the vocabulary decreases. For the government data, the total number of words increased by a factor of 1.227 and the English translation vocabulary size is only 0.34 (13042 words) the size of the Norwegian vocabulary. For the subtitles data, the total number of words increased by a factor of 1.161 and the English translation vocabulary size (46232 words) is only 0.644 the size of the Norwegian vocabulary. This can be explained by multiple Norwegian words being translated to the same English word. Additionally, the English translation vocabulary shares only 8582 words. The book corpus shares 89.4% of its vocabulary with the combined vocabulary, which account for 97.93% of the total words in the whole book corpus. The DIY restaurant shares 88.24% of its vocabulary with the combined vocabulary which is 93.69% of the total words in the DIY dataset. Based on this we can infer that the performance on the DIY dataset will perform worse than the book dataset.

English validation data analysis - vocabulary			
Data	**Vocab size**	**vocab shared /w combined**	**shared vocab % of total corpus**
Book	4 132	89.4%	97.93%
DIY restaurant	626	88.24%	93.69%

Table 3: Data analysis of English validation and test data with focus on vocabulary sharing with the English combined data.

5 Model

The overall architecture of our model follows a regular sequence-to-sequence network as depicted in Figure 1. A standard architecture format where the Norwegian input tokens are the source, and the English tokens are the target input. The beginning-of-sentence (BOS) and end-of-sentence (EOS) tokens are both used in the encoder part, but for the decoder only the BOS token is used for the target input and only the EOS token is used for the target predictions. An overview of both the encoder and decoder part of our model can be seen in Figure 2. The encoder is built up by a multi head attention (self attention) module followed by a normalization module, feed forward layer, normalization module and then a dropout module before it's fed to the decoder. A multi head attention (self attention) layer, normalization module, multi head attention (cross attention), normalization module, feed forward layer, normalization module and dropout module constitute the decoder part of our model. Thus, we are following the encoder-decoder architecture (Vaswani et al., 2017), but with the improvements (Nguyen and Salazar, 2019), (Press et al., 2021), (Shazeer, 2020) as mentioned.

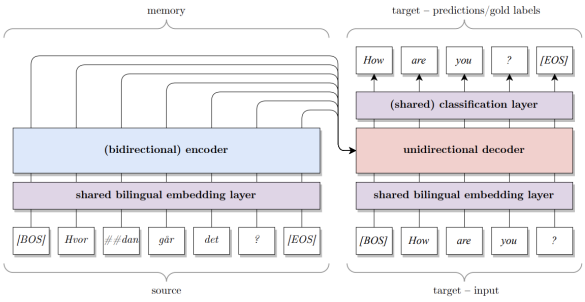


Figure 1: Simplistic diagram of the encoder-decoder architecture.

rparameters. Hidden size = embedding size = 300, number of encoding and decoding layers = 6, number of attention heads = 6, batch size = 1024, learning rate = 0.001, epochs = 100, max sentence length = 256, dropout = 0.1, weight decay = 0.1. We use the AdamW optimizer and cosine decay scheduler. The learning rate is chosen after some try and fail runs. The rest are set based on practical purposes as the dataset size we operate with is very small compared to the one used in the Vaswani paper (Vaswani et al., 2017). Our model has the "pre-norm" variant from applying layer-normalization to the input of every sublayer. The "position-infused attention" idea (Press et al., 2021) is applied, adding position embeddings to keys and queries, but not the values, which makes the values position independent and representations from previous subsequences can seamlessly be processed. Lastly, our model utilizes the gated linear unit (GLU) non-linearity (Shazeer, 2020), an activation function that allows the network to decide how much information should flow through a path. The GLU is inspired by the gates of the LSTM cell. Our model also has a bidirectional encoder and a unidirectional decoder.

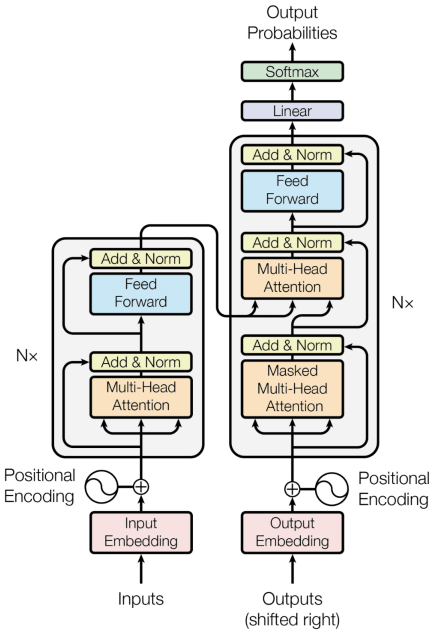


Figure 2: The encoder-decoder structure of the Transformer architecture (Vaswani et al., 2017).

6 Results

We trained three models for each respective dataset as described in previous sections. From table 4 it is evident that the models trained on the government and subtitles corpuses performs the best at their own validation set, as expected. The model trained on the government corpus evaluated on the government and subtitles validation set has BLEU scores of 42.77 and 8.93, respectively. The model trained on the subtitles corpus evaluated on the government and subtitles validation set has 12.83 and 32.11 BLEU scores, respectively. With BLEU scores of 42.96 and and 32.0 we see that the combined model achieves almost identical BLEU scores on the government validation set and subtitles validation set as the government model and subtitles model does, meaning that adding the specific data domain doesn't weaken our predictions, but rather enriches the model compared to the two other models. This was evident when we evaluated the models on the Book dataset, which we used as our main decision point for choosing the best model. We observed that the combined model outperformed the other two models with a BLEU score of 18.11, whereas the government model only had a BLEU score of 6.6 and the subtitles model has a score of 14.61. A similar pattern is evident in our own DIY dataset where the government model had a BLEU score of 8.19, subtitles model had 17.42 and the combined model had 22.47. We believe that the generally low BLEU scores on the Book and DIY dataset is due to small training dataset and varying translation quality in both the subtitles gold labels and especially in the DIY dataset. We kept this in mind for the qualitative part of our evaluation.

	Validation set			
Models	**Government**	**Subtitles**	**Book**	**DIY**
Government	42.77	8.93	6.60	8.19
Subtitles	12.83	32.11	14.61	17.42
Combined	42.96	32.0	18.11	22.47

Table 4: BLEU scores

Curiously we found that the shared vocabulary between the combined model and the book dataset is 95.52% and the shared vocabulary between the combined model and the DIY dataset is 84.7%, meaning though the book dataset has a higher shared percentage, it performs worse than the DIY dataset, but note that this could just be down to the number of words not shared as the missing 4.48% is around 1 514 words in the Book dataset whereas the missing 15.3% from the DIY dataset is just 196 words. However, we saw a stronger link to performance when looking at data not seen during training. When predicting on the Book dataset, the

subtitles model had not seen 21,46% of the Norwegian words and 11,88% of the English words during training. Also, 11,72% of the predicted words from the model had not been seen. For the government model the percentages are 54,87% of the Norwegian words, 44,89% of the English words and 24,7% of predictions made. For the combined model the percentages are 19,99% of the Norwegian words, 10,6% of the English words and 8,37% of the predictions made. The inverse relationship between BLEU score and the percentage values seems to hold here as the combined model has the lowest percentages and the highest scores. A similar pattern is found when evaluating on the DIY dataset with 23.54% of Norwegian words, 11.82% of English words and 12.52% of predicted words for the combined model. 25.49% of Norwegian words, 12.78% of English words and 15.99% of predicted words for the subtitles model. 50.07% of Norwegian words, 43.29% of English words and 19.6% of predicted words for the government model.

6.1 Analyzing examples where different models predict differently

1. Book example:

   Input: Jeg innrømmer, kjære venn, at jeg står i stor gjeld til Dem.
   Target: I confess, my dear fellow, that I am very much in your debt.
   Pred gov: I admitte, cheape walls, that I are largely liable to them.
   Pred sub: I admit, my dear friend, that I owe you a lot of debt.
   Pred comb: I admit, my dear friend, that I am largely in debt to you.

   In example 1 we can see that the model trained on the combined corpus and the subtitle corpus predicts sentences that largely makes sense, whereas the model trained on the government corpus does not create a sentence that make fully sense. This is most likely due to the fact that the word "friend" nor its Norwegian counterpart "venn" does not appear in the training corpus.

2. DIY example:

   Input: Dinner restaurant har servert mat fra det Szechuanske- og kantonesiske kjøkkenet i hjertet av Oslo siden 1989.
   Target: Dinner restaurant has served food from the Szechuan and Cantonese cuisine since 1989 in the heart of Oslo.
   Pred gov: Your restaurant has food from the Skaweruan and Central Economic body at the heart of Oslo since 1989.
   Pred sub: Your restaurant has served food from the Szechuan kitchen in the heart of Oslo since 1989.
   Pred comb: Dinner restaurant has served food from the Szechuan and Cantonese kitchen in the heart of Oslo since 1989.

   For example 2 we can see that the combined model almost recreates the sentence fully while the two other models struggle, especially the model trained on the government corpus. An interesting detail is that when the model trained on the subtitles corpus and the model trained on the government corpus is evaulated by themselves, they both fail to detect the name of the restaurant in the beginning, however when the corpuses is combined, the model picks up on it.

3. Gov example:

   Input: Årsaker som opp- gis av de ulike fylkeskommunene, er:
   Target: Reasons given by the various county municipalities include:.
   Pred gov: Causes given by the different county authorities are :
   Pred sub: Causes as upside - hostages of the different county Communes, are :
   Pred comb: Causes that were displayed by the various county municipalities are :

4. Sub example:

   Input: Han er faktisk ganske sjenert.
   Target: He's actually quite shy.
   Pred gov: He is actually found to be quite source.
   Pred sub: Actually, he's pretty shy.
   Pred comb: He's actually quite shy.

6.2 Analyzing examples

Sentence analysis on book validation and DIY dataset using the combined.

Reasons the word is wrong:

- Underline: Correct translation, but wrong context.
- **Bold**: Norwegian or English input word is not in training data.

- *Italic*: Translation captures parts of the word due to a sunword token.
- *stars around*: Model makes up a new word; the word predicted is not in the English training data
- Mix: a mix of reasons can occur. For example: **Vineddik*saus*** → **wind dictix *sauce***

  "Vineddiksaus" as a word is not in the Norwegian training data, but the model recognizes "saus" as a subword and translates it correctly to sauce.

1. DIY example:

   Input: Sprøstekt finstrimlet oksekjøtt, serveres med en klassisk **vineddiksaus**.

   Target: Crispy beef strips in a classic vinegar sauce.

   Pred: The **crazy *finstrimum*** of **bulk - eye** is served using a classic **wind *dictix*** ***sauce***.

   In example 1, there are a lot of domain specific words, which affect the translation. The Norwegian words "sprøstekt", "finstrimlet", "oksekjøtt", and "vineddiksaus" are not in the Norwegian training data. "Sprøstekt" is translated to "crazy", which most likely comes from the subword token "sprø" which can be translated to "crazy".. "Finstrimlet" is translated to "finstrimum", which is a made-up word by the model, a result of non-exposure to the original word during training. "Vineddiksaus" is translated to "wind dictix sauce", the model catches the subword token "saus" and correctly translate it to "sauce", whereas the rest of the word is made up.

2. DIY example:

   Input: **Koldtbord** og **snitter** er kanskje fortsatt det tradisjonelle valget I dåp, konfirmasjon og konferanser.

   Target: Sandwiches and cold cuts are still considered to be the traditional choice for christenings, confirmations and conferences.

   Pred: **Cold table** and **averages** may still be the traditional election in baptism, confirmation and conferences.

3. DIY example:

   Input: Vårruller (en med kylling og en med laks) og en **Sumo ebi** (**scampi**) tempura servert med **chilimajo**, sweet chili, **teriyakisaus** og salat.

   Target: Spring rolls (one chicken and one salmon) and a Sumo **ebi** (**scampi**) tempura served with chili mayo, sweet chili, **teriyaki** sauce and salad.

   Pred: Spring rolls (one with chicken and one with salmon) and a **Sumo Ebi** (**scampi**) tempura served with ***chilia***, ***exhi***, **teriyakisaus** and salad.

4. Book example:

   Input: Delen som vendte mot oss dannet en mørk **fjellskrent**, med **bregner** og bringebær voksende i **kløftene**.

   Target: The face which was turned towards us formed a dark cliff, with **ferns** and brambles growing in its niches.

   Pred: The part that turned against us was forming a dark **rock crew**, with **brilliant** and berry growing in the ***raves***.

5. Book example:

   Input: "Denne **moen** er et vidunderlig sted," sa han og så utover det **bølgende** landskapet og de lange grønne **forhøyningene** med de **takkete granittkammene**, som reiste seg som fantastiske, **skummende** sjøer.

   Target: "It is a wonderful place, the moor," said he, looking around over the **undulating** downs, long green rollers, with **crests** of jagged granite foaming up into fantastic surges.

   Pred: "This **foam** is a wonderful place," he said and then beyond the **waterful** landscape and the long green **higher highers** with the **granitism camp**, who travelled like a wonderful, **scary** sea.

7 Conclusion and future work

Our results showed that it is necessary for the model to have seen the words in the same text domain it should translate during its training phase in order to create accurate sentences. The words should also have been used in a similar sentence structure as the one provided to the model for inference. This matches with expectations as within machine learning, a model will do poorly on data it has never seen before. Our observations show that the model will simply start guessing when it is given words that it has never seen before. For the menu translation use-case it meant that it would start guessing on the ingredients the items in the menu was composed of.

Regularization techniques are commonly used to increase a model's performance on out-of-domain data through generalization. Future work could involve looking into how regularization techniques could help the model guess on words which are not included in the training corpus. Our results also show that the frequency of words in the training corpus matters for the models predictions using said words. An analysis of how regularization techniques help the model with prediction words of low

frequency in the training corpus would also be of interest.

References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Jiang, Z., Xu, F. F., Araki, J., and Neubig, G. (2020). How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Lison, P. and Tiedemann, J. (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).

Nguyen, T. Q. and Salazar, J. (2019). Transformers without tears: Improving the normalization of self-attention.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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

Press, O., Smith, N. A., and Lewis, M. (2021). Shortformer: Better language modeling using shorter inputs. 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 5493–5505, Online. Association for Computational Linguistics.

Provilkov, I., Emelianenko, D., and Voita, E. (2020). Bpe-dropout: Simple and effective subword regularization.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shazeer, N. (2020). Glu variants improve transformer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Targeted Sentiment Analysis for Norwegian: An Experimental Study of Sentiment Intensity

Lilja Charlotte Storset
University of Oslo
Department of Informatics
liljacs@ifi.uio.no

Fredrik Aas Andreassen
University of Oslo
Department of Informatics
fredaan@ifi.uio.no

Abstract

Targeted sentiment analysis attempts to extract sentiment targets and the polarity directed towards them, and is the task that we will experiment with using a simpler version of the NoReC$_{fine}$ dataset (Øvrelid et al., 2020), named NoReC$_{TSA}$. We test four versions of the LTG NorBERT3 model to classify the polarity of sentiment targets, in which the last three versions are fine-tuned. In addition, we experiment with sentiment intensity to see how the performance differs. Our experiments show that NorBERT3 can be fine-tuned to receive better results, and that adding sentiment intensity complicates the task significantly. Thus, there is much room for improvement on that specific area.

1 Introduction

Sentiment analysis (SA) is an active field within natural language processing, which mainly involves analyzing peoples sentiment, emotions, opinions and attitudes towards several entities and their attributes. The term sentiment analysis has become an umbrella term for many related sentiment tasks, such as opinion mining, opinion analysis, emotion analysis, opinion extraction, and more. (**sentiment_bok**). Sentiment analysis has been one of the most active research areas of the last 10 years, one of the reasons being that the field has a wide range of applications. Some of many domains, are social media monitoring, customer support managing, customer feedback analysis, stock market prediction, in addition to sentiment analysis related to politics, such as public opinions.

The three most spoken-about levels of granularity for sentiment analysis are document level, sentence level, and aspect level. However, aspect level has not been used in SA for Norwegian language, which is why we instead include the term *structured sentiment analysis* (SSA) as the finest level. This is a slightly more fine-grained level than aspect level, because it analyzes the polar expressions directly, rather than categorizing the polar expressions into more general categories, like aspect level does. **Targeted sentiment analysis** (TSA) can be viewed as a level between sentence level and structured level sentiment analysis, as it has more attributes than sentence level, but fewer attributes than SSA. TSA keeps track of the **target** of the sentence, that is, the entity that has some polarity directed towards it, and the second attribute is the **polar expression**. The polar expression is the expression in a sentence that contains some polar opinion about the target, e.g. a positive or a negative opinion. There have been used many names to refer to targeted sentiment analysis, including aspect-based sentiment analysis, but we recognize that as something different, as described earlier in this section. The term "target" is a precise term to describe the task, as it analyzes the polar expression directed towards a target in a sentence. The Figures 1 and 2 show respectively how a sentence would be annotated in a TSA- and SSA style. The main dataset used in this task is the NoReC$_{TSA}$ dataset, which is based on the more fine-grained version NoReC$_{fine}$ (Øvrelid et al., 2020). As we experiment with sentences annotated both with and without sentiment intensity, we will discuss both NoReC$_{TSA}$ and NoReC$_{fine}$, in which the last mentioned dataset contains annotations *with* sentiment intensity.

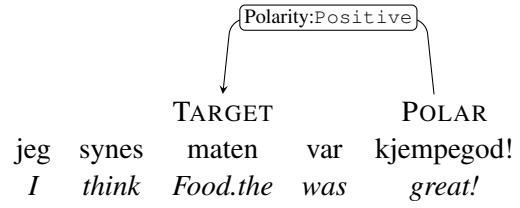


Figure 1: TSA-annotation of 'Jeg synes maten var kjempegod!' (transl. 'I think the food was great!'), containing a polar expression and a target.

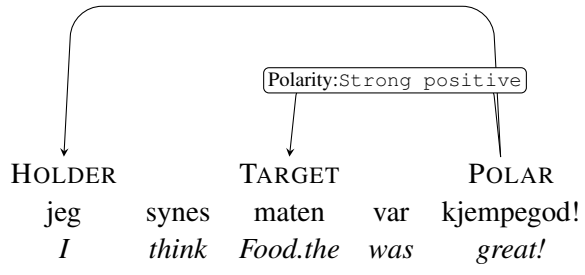


Figure 2: SSA-annotation of 'Jeg synes maten var kjempegod!' (transl. 'I think the food was great!'), containing the holder, target and the polar expression, as well as the sentiment intensity.

As seen in Figure 2, the SSA-annotation, in addition to the polar expression and target of the sentence, also contains the *holder* of the sentence, which is the entity that *holds* the opinion. It also contains *intensity* for the polar expression, showing the grade of positivity for the sentence, which in this case is *strong*. Figure 1 shows the TSA-annotation of the same sentence, containing only the target of the sentence, and the polar expression. The polar expression does not contain intensity like SSA, but instead has a simpler annotation showing only if the polar expression is of positive or negative sentiment. The TSA-annotation shows that the polar expression *kjempegod!*, 'great!' has been assigned positive polarity, because *kjempegod!* indicates that the food was good. The SSA-annotation shows a more detailed analysis of *kjempegod!*, where the polar expression has been assigned *strong positive* sentiment, as *kjempegod!* indicates that the food was better than standard. The two datasets will be further discussed in Section 3.

2 Related work

There are not many publicly available datasets that focus on the task of TSA as we define it, but the most widely used for similar tasks, is the SemEval 2014 dataset (Pontiki et al., 2014). This dataset follows the aspect-based annotation, i.e. it contains categories that represent the target, rather than the target itself. The data contains online reviews for restaurants and laptops, an are annotated with four possible polar categories; *positive, negative, neutral* and *conflict*. In addition to SemEval 2014, there are two datasets containing tweets. The first dataset, collected by Dong et al. contains "keywords" which are names of celebrities, companies and products, and the labeled polarity directed towards them. The dataset has three polar categories; *positive, negative* and *neutral*. This dataset is probably the one that is closest to the TSA annotation of NoReC$_{TSA}$. The second Twitter collection (Hu et al., 2019) contains 2350 English and 7105 Spanish tweets, where each entity is categorized as either *Person* or *Organization*. Therefore, this is more similar to a aspect-based approach. An overview of the distribution of attributes between these datasets can be found in Table 1. All of the attribute distributions that were found for NoReC$_{TSA}$, were not provided for all of the other datasets, which is why many of the rows contain "-". The total of sentences (tweets for the Twitter datasets) and distribution of polar expressions can however be compared across almost all of the datasets, apart from the Spanish/English Twitter dataset.

When it comes to modeling of targeted sentiment analysis, which only focuses on extracting the sentiment targets (task ii) and classifying the polarity directed towards them (task iv), recent shared tasks have been proposed to similar tasks. Barnes et al. sums up the work that has been done to solve these tasks: The shared tasks for 2014 (Pontiki et al., 2014), 2015 (Pontiki et al., 2015) and 2016 (Pontiki et al., 2016) involved Aspect-Based Sentiment analysis (ABSA) (as mentioned in Section 1), which in addition to the task of ABSA, also included extraction of targets and polarity classification. In solving these tasks, joint approaches have been tried, but multitask models can perform even better. Lastly, pre-trained language models (Devlin et al., 2019) can also improve the ABSA data. (Barnes et al., 2021)

3 Dataset

The NoReC$_{TSA}$ dataset is a reduced dataset in respect to the number of attributes, derived from the NoReC$_{fine}$ (Øvrelid et al., 2020). NoReC$_{fine}$ is a highly relevant dataset for Norwegian structured sentiment analysis, which is a more fine-grained analysis than the NoReC$_{TSA}$ dataset contains. NoReC$_{fine}$ is the only dataset for Norwegian which is annotated on a fine-grained level, and is thus the first of its kind. The underlying texts are taken from the Norwegian Review Corpus (NoReC) (Velldal et al., 2018), which is a collection of professionally authored reviews from a wide variety of domains, including literature, video games, music, movies, TV-series, restaurants, products, etc. NoReC$_{fine}$ contains 11 437 sentences across more than 400 reviews and 10 different thematic cat-

egories. The sentences are annotated to include spans of polar expressions with the corresponding targets, holders and polarity. NoReC$_{fine}$ also keeps track of the intensity of the polarity on a three point scale, for either positive or negative sentiment. The intensity is either *standard*, *slight*, or *strong*. What differs NoReC$_{TSA}$ from NoReC$_{fine}$, is that NoReC$_{TSA}$ only keeps track of the polarity for a given target, without the intensity, which makes the task of sentiment analysis simpler. For this task, we will experiment both with and without polarity intensity. NoReC$_{fine}$ counts 1732 sentiment targets with more than one opinion towards it, where 402 of them have been assigned both positive and negative polarity. In such cases, the polarity with the highest intensity has been chosen for the NoReC$_{TSA}$ dataset. (2022)

Statistics of the distribution of sentences, targets, polar expressions and polarity in the NoReC$_{TSA}$ dataset can be seen in Table 1, along with the statistics for the rest of the datasets discussed in Section 2. As shown in the table, NoReC$_{TSA}$ contains a quite high amount of sentences, and is thus one of the largest datasets for TSA. It is also worth mentioning that the underlying dataset, NoReC$_{fine}$, also is one of the biggest datsets for SSA, even across languages. The most relevant attribute in this article, will be the "polarity"-attribute, in which NoReC$_{TSA}$ contains 5684 positive polarity attributes, and 2756 negative. As we can read of Table 1, there is an imbalance between the frequencies of positive vs. negative polarity, which will be further touched upon through out the article.

The TSA repository (2022) provides a conversion script, that simplifies the annotation, as well as converting it to CoNLL format, provided with BIO-tags and polarity. BIO stands respectfully for 'Beginning', 'Inside' and 'Outside', and indicates if the polar expression occurs at the beginning of a polar expression, inside or outside the polar expression. An example of a sentence in such format can be seen in Figure 3 (2022). The same figure also shows a case where there are two polar expressions with two different polarities originally assigned to each word, but has been provided with only one for the NoReC$_{TSA}$ dataset. *brodd*, 'sting', was originally assigned *strong* positive polarity, while *bismak*, 'bad taste', was assigned *standard* negative sentiment. Therefore, positive polarity was chosen.

```
#sent_id=004702-02-01
Forviklingskomedie  B-targ-Positive
med     O
brodd   O
og      O
bismak  O
.       O
```

Figure 3: CoNLL formatting of a sentence in TSA dataset

4 Baseline Model

NorBERT 3

The provided code for the baseline model, provided several different options for models to run, in which we chose the NorBERT 3 model. NorBERT 3 is the latest version of the NorBERT models created by the Language Technology group (LTG) in the Department of Informatics, University of Oslo. "ltg/norbert3-base · Hugging Face", n.d.

The model comes in four sizes; xs, small, base and large. The xs model has 15 million parameters, small has 40 million parameters, base has 123 million parameters, and lastly, large has 323 million parameters. The NorBERT models are large-scale contextualized language models for Norwegian language and have transformer-based architectures.

4.1 Baseline parameters

We tested with both NorBERT 3 xs, small and base, but large was unfortunately too much for the memory to handle. Therefore NorBERT 3 base was the model that we continued using, which gave better results than the smaller versions. The original baseline code that was provided kept all the labels, meaning all BIO-tags for both sentiment types (negative and positive) were included. These settings resulted in five labels for NoReC$_{TSA}$ without intensity for polarity, and thirteen labels for the dataset with labeled intensity. We wanted to keep the focus on the classification of sentiment for the task, rather than both sentiment and BIO-classification, which is why we decided to remove the BIO-tags, and thus only keeping the sentimental tags for our task. After having done these alternations, we saw an increase of about 10% in F1-score, which makes sense given the decreased number of labels.

For our first parameters, the batch size was set to 32, and the number of epochs was set to 10. First we tested with the original, unmodified TSA dataset, which means that the only polarity labels are *positive* and *negative*. The NoReC$_{TSA}$ repos-

		#Sentences/Tweets	#Targets	#Polar exp.	Polarity		
					#Pos.	#Neu.	#Neg.
NoReC$_{TSA}$	train	8634	6778	8448	6885	0	3030
	dev	1531	1152	1432	1205	0	500
	test	1272	993	1235	1097	0	373
	total	**11.437**	**8923**	**11.115**	**9187**	0	**3903**
SemEval 2014	train	6086	-	-	2164	633	805
	dev	-	-	-	-	-	-
	test	1600	-	-	728	196	196
	total	**7686**	-	-	**2892**	**829**	**1001**
Twitter Keywords	train	6248	-	-	-	-	-
	dev	-	-	-	-	-	-
	test	692	-	-	-	-	-
	total	**6940**	-	-	**1735**	**3470**	**1735**
Twitter Spanish/English	train	-	-	-	-	-	-
	dev	-	-	-	-	-	-
	test	-	-	-	-	-	-
	total	**9455**	-	-	-	-	-

Table 1: Statistics for the TSA datasets. "-" means that the specific information was not provided by the creators.

itory ("Convert NoReCfine sentiment targets to CoNLL", 2022) provided the code which converted the original NoReC$_{fine}$ data to NoReC$_{TSA}$, and so we later changed the script by adding intensity. We ran the same baseline models with this modification as well, which means that the labels for this dataset is *positive standard*, *positive slight*, *positive strong*, and for the negative polarity, *negative standard*, *negative slight* and *negative strong*. This resulted in an extra four labels, which also meant that the number of examples for each label was quite reduced. For that reason, it was an exciting await for the results, which can further be viewed in Section 5.2. First, we discuss the results for the original NoReC$_{TSA}$ dataset in Section 5.1.

5 Results: Baseline

Every table that provides model results, will have a title describing the batch size and number of epochs. For instance, out first results in Table 2, has the title "Norbert_32_10", which means that the model has a batch size of 32, and 10 epochs.

The baseline model for NoReC$_{TSA}$ without intensity retrieved a fairly high accuracy of 95.2%, but because the dataset is imbalanced in the types of polarity, as seen in Table 1, we will for the most part look at the more relevant metric, F1-score. The NoReC$_{TSA}$ development set contains 1205 examples with positive polarity, and 500 examples with negative examples, which is less than half the size of the positive examples. The F1-score for the baseline model is therefore much lower than accuracy, and retrieved a score of 52%.

5.1 Original NoReC$_{TSA}$ dataset

As we can tell by Table 2, all of the metrics are higher for the positive label than for the negative label because of the data imbalance, which will also be observed for the rest of the results.

5.2 NoReC$_{TSA}$ with Intensity

For the NoReC$_{TSA}$ version with intensity, we edited the dataset to keep the polarity intensities from the original NoReC$_{fine}$ dataset, which adds three degrees of intensity: `standard`, `slight`, and `strong`. Further, we ran the same baseline model as in Section 5.1. From the results in Table 3, we can see that the accuracy is still very high, with a score of 94.5%, but we still have to bear in mind the unbalanced dataset, which is why we turn to the F1-score. As expected, the F1-score dropped quite a lot, and retrieved a score of 34.57%. That is a decrease of 18%, from 52%. This is not very surprising, given that there are four more labels to classify, and thus less examples for each label.

The support for each class, which can be seen in

Baseline: NorBERT_32_10	precision	recall	f1-score	support
targ-positive	0.77	0.45	0.57	1205
targ-negative	0.52	0.33	0.40	500
micro avg.	0.69	0.42	**0.52**	

Table 2: Results for the baseline model with a batch size of 32 and 10 epochs.

Baseline with Intensity: NorBERT_32_10	precision	recall	f1-score	support
Negative_standard	0.43	0.27	0.33	377
Negative_slight	0.0	0.0	0.0	64
Negative_strong	0.0	0.0	0.0	59
Positive_standard	0.62	0.33	0.43	900
Positive_slight	0.0	0.0	0.0	**36**
Positive_strong	0.52	0.33	0.40	269
micro avg.	0.5283	0.2569	**0.3457**	

Table 3: Results for the baseline model *with intensity*, with a batch size of 32 and 10 epochs.

Table 3, is quite important when understanding the retrieved metrics for the dataset with intensity. We remember from earlier that the number of negative examples are much lower than the positive examples, and with the added intensity, the number of examples for especially negative polarity, has lowered drastically. E.g. there are only 59 examples for *negative strong*, compared to 269 examples for *positive strong*. Even lower is the number of examples for *positive slight*, which only has 36. Overall, there is a very low number of examples for anything other than the *standard* examples. There are simply not enough examples for these categories, which results in some of the scores being 0.0.

6 Fine-tuning the model

When trying to improve the unmodified NorBERT 3 baseline model, we started with experimenting with the batch size and number of epochs. As read, our baseline model had a batch size of 32 and a number of 10 epochs. We did two modifications of this model, where the second modification had the same number of epochs, but an increased batch size of 64. Our second modification had a batch size of 64, and an increased 30 number of epochs. Our last modification included adding a linear layer on top of the BERT layer. The results for these models can be found in the next section (7).

7 Results: Fine-Tuned Models

7.1 original TSA

For the sake of simplicity, we do not show the support for each class in the upcoming tables, as the number of examples remain the same, both *with* and *without* intensity. We also only include the F1-score for each of the models, as this is the most relevant metric, but we will include all of the metrics when showing the results for the final test set in Section 8. Table 4 shows a comparison of the results from all of the modified models without intensity, including the baseline model, where the second column represents the first modification of our model. The first modification contained a batch size of 64 and 10 epochs. As seen in the table, our F1-score improved from 52% to 61% when doubling the batch size. This is an increase of 8%, which shows that batch size is a rewarding parameter to tune. For our next modification, we kept the batch size of 64, but increased the number of epochs from 10 to 30. As seen in Table 4, the F1-score actually dropped down to 52%, which is a decrease of 9%. This leads us to believe that the model was overfitted with 30 epochs. We do, however, have one last modification, which is adding a linear layer on top of the BERT-model. When adding the layer, we kept the batch size of 64, but tested with both 10 and 30 epochs to see if the model would produce an overfitted result, even

with the linear layer. It actually did not. In fact, our last modification did better with 30 epochs than with 10. With these settings, the model with 10 epochs retrieved an F1-score of 55%, while the model with 30 epochs gave back a score of 61%, in which the results for the best model can be seen in the same table (4) along with the other results.

7.2 TSA with intensity

So far, we have seen the results from several modifications of the NorBERT 3 base model, tested on the unmodified TSA dataset. Now, we present the results for the NoReC$_{TSA}$ dataset *with intensity*, using the best settings for the model, but trained and evaluated on the dataset with added sentiment intensity. The results can be seen in Table 5.

Best Model with Intensity: NorBERT_64_30	f1-score
Negative_standard	0.38
Negative_slight	0.0
Negative_strong	0.0
Positive_standard	0.52
Positive_slight	0.0
Positive_strong	0.25
micro avg.	**0.43**

Table 5: Results for the best model *with intensity*, with a batch size of 64, 30 epochs and a linear layer.

As seen in Table 5, the categories that retrieved scores of 0 for the baseline model, are still 0, which leads us to believe that these categories need more examples in order to improve the most, all though there is definitely possible to experiment with more fine-tuning. All though the zero-scores stayed the same, the F1-score improved from 34% to 43%, giving an increase of 9%, which is the exact same increase as for the unmodified dataset. We are satisfied with the results improving for both the unmodified NoReC$_{TSA}$ dataset, and the dataset with intensity!

8 Final results on test set

Finally, we procede to present the results of the test set, using the best settings from Section 7. The results are retrieved using the same two models from the same section, trained on the train data from the original NoReC$_{TSA}$ dataset and the train data with intensity. The results for the unmodified data can be seen in Table 6. Here, we again provide the support for each label, as the number of examples are a bit decreased for the test set. We also provide a confusion matrix for each granularity, showing where the models did the best, and where it struggled the most.

NoReC$_{TSA}$ without intensity

We are pleased to see that the F1-score barely dropped with 1%, from 61% to 60%, even though there are fewer examples. Compared to the very first baseline model run on the development set, this is an increase of 8%. Figure 4 is a confusion matrix showing where the model performed best, and where it struggled the most.

From the confusion matrix, we can clearly see that the model classifies the positive class much better than the negative, and is most likely due to there being more than twice the amount of the positive labels than the negative labels.

True:

```
#sent_id=201344-14-01
Den          O
opplevde     O
kvaliteten   B-targ-Positive
er           O
god          O
```

Predicted:

```
#sent_id=201344-14-01
Den          B-targ-Positive
opplevde     I-targ-Positive
kvaliteten   I-targ-Positive
er           O
god          O
```

Figure 5: A sample showing how the model classifies a whole noun phrase, 'Den opplevde kvaliteten' (transl. 'The experienced quality'), as a positive target instead of only the head of the noun phrase, 'kvaliteten'.

NoReC$_{TSA}$ with intensity

We can see in Table 7 that the F1-score dropped more for the test set with intensity, than the test set without intensity. While the F1-score for the unmodified test set dropped by only 1%, the F1-score for the dataset with intensity dropped by 7%, from 43% to 36%. This shows that the existing examples for the intensity labeled dataset was very much needed for the score to maintain, while the binary dataset had enough to maintain its score. The zero-valued scores remains the same for the

Comparison: All Models without Intensity	Baseline	Modified #1	Modified #2	Modified #3
Positive	0.57	0.65	0.60	0.66
Negative	0.40	0.48	0.33	0.47
micro avg. F1-score	**0.52**	**0.61**	**0.52**	**0.61**

Table 4: Comparison of all models without intensity. *Baseline* batch size:32, epochs:10, *Modified #1* batch size:64, epochs:10, *Modified #2* batch size: 64, epochs: 30, *Modified #3* batch size:63, epochs:10, linear layer.

Final Model on Test: NorBERT_64_30	precision	recall	f1-score	support
Positive	0.70	0.61	0.65	1097
Negative	0.48	0.38	0.42	373
micro avg.	0.65	0.55	**0.60**	

Table 6: Results with the best model on the original TSA test set.

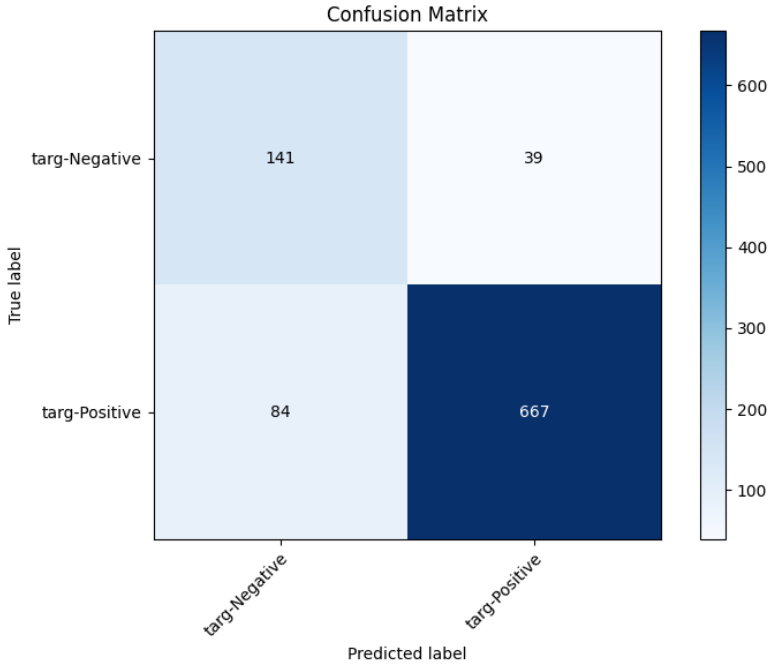


Figure 4: Confusion matrix showing errors for the final model on test set *without intensity*

test set, and can additionally be seen very clearly in the associated confusion matrix in Figure 6.

In addition to showing difficulties with the zero-scored labels, the confusion matrix in Figure 6 also clearly shows, like for the unmodified dataset, the advantage of the positive class having more examples than for the negative, especially for the standard intensity, which also does slightly better for the negative class.

9 Conclusion

We have seen that it is possible to fine-tune the NorBERT3 model to retrieve better results for the NoReC$_{TSA}$ dataset, with and without polarity intensity. We also know that one can fine-tune much more than we have done, e.g. by adding more and/or different layers, adjust dropout, as well as experimenting more with different epoch and batch size combinations. Based on the result we have seen, we believe that such modifications would improve the model for this task even more. Further,

Final model on test w/ intensity: NorBERT_64_30	precision	recall	f1-score	support
Negative_standard	0.37	0.37	0.37	280
Negative_slight	0.0	0.0	0.0	50
Negative_strong	0.0	0.0	0.0	43
Positive_standard	0.43	0.47	0.45	655
Positive_slight	0.0	0.0	0.0	50
Positive_strong	0.59	0.16	0.25	378
micro avg.	0.43	0.32	**0.36**	

Table 7: Results for the final model *with intensity*

Figure 6: Confusion matrix showing errors for the final model on the test set *with intensity*

we saw that the models with added polarity intensity gave much lower results than the models with binary polarity, and particularly struggled a lot with the low-supported classes. This also reflected in the final test scores, which decreased more for the dataset with added polarity intensity, than for the one without. As the results are now, we do not believe that the model with added sentiment intensity would be useful for prediction of such tasks. Therefore, we are curious to know if data augmentation, i.e. automatically create more equivalent data could improve these results, perhaps even more than experimenting with parameters of the models.

References

Barnes, J., Kurtz, R., Oepen, S., Øvrelid, L., & Velldal, E. (2021). Structured Sentiment Analysis as Dependency Graph Parsing. *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)*, 3387–3402. https://doi.org/10.18653/v1/2021.acl-long.263

Convert NoReCfine sentiment targets to CoNLL [original-date: 2022-08-12T09:20:25Z].

(2022). Retrieved May 15, 2023, from https://github.com/ltgoslo/norec_tsa

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *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)*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Dong, L., Wei, F., Tan, C., Tang, D., Zhou, M., & Xu, K. (2014). Adaptive recursive neural network for target-dependent twitter sentiment classification. *The 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hu, M., Peng, Y., Huang, Z., Li, D., & Lv, Y. (2019). Open-Domain Targeted Sentiment Analysis via Span-Based Extraction and Classification. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 537–546. https://doi.org/10.18653/v1/P19-1051

Ltg/norbert3-base · Hugging Face. (n.d.). Retrieved May 16, 2023, from https://huggingface.co/ltg/norbert3-base

Øvrelid, L., Mæhlum, P., Barnes, J., & Velldal, E. (2020). A Fine-grained Sentiment Dataset for Norwegian. *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 5025–5033. Retrieved March 22, 2023, from https://aclanthology.org/2020.lrec-1.618

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., AL-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N., Kotelnikov, E., Bel, N., Jiménez-Zafra, S. M., & Eryiğit, G. (2016). SemEval-2016 Task 5: Aspect Based Sentiment Analysis. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 19–30. https://doi.org/10.18653/v1/S16-1002

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., & Androutsopoulos, I. (2015). SemEval-2015 Task 12: Aspect Based Sentiment Analysis. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 486–495. https://doi.org/10.18653/v1/S15-2082

Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., & Manandhar, S. (2014). SemEval-2014 Task 4: Aspect Based Sentiment Analysis. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 27–35. https://doi.org/10.3115/v1/S14-2004

Velldal, E., Øvrelid, L., Bergem, E. A., Stadsnes, C., Touileb, S., & Jørgensen, F. (2018). NoReC: The Norwegian Review Corpus. *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Retrieved May 16, 2023, from https://aclanthology.org/L18-1661

Benchmarking Targeted Sentiment Analysis Models

Roman Macháček
romanma@uio.no

Abstract

Targeted Sentiment Analysis (TSA) deals with extracting sentiment targets and assigning sentiment polarity to them. TSA can be thought of as a combination of two tasks, Named Entity Recognition (NER) and Sentiment Assignment (SA) for Entities. Sentiment Assignment can include not only polarity (Positive, Negative) but also Intensity (1-3), as in Norwegian NoReC$_{tsa}$ dataset. The goal of this paper is to thoroughly evaluate and train models for sentiment analysis, based on factors such as category and zero, few-shot learning. Word analysis is carried out, to see how the models perform on the word level, together with hypothesis testing about the elements that may effect the sentiment assignment. Custom edge-case sentences were developed, to identify, if the model has bias and can understand logical statements, such as negation and nesting. Results showed the shortcomings of the trained models and further directions for improvements were provided.

1 Introduction

Internet and its usage has become one of the most prevalent things in many lives. Millions of people write reviews, feedback, descriptions, blogs and messages every day. It is practically unlimited, ever increasing source of natural language data with great usage potential. To gain useful knowledge from this source, extraction of the data is required. Extracted data is then used to find patterns in order to make predictions. Many methods exist, while currently, the most prevalent method is Deep Learning (Bishop, 2006). There are hundreds of applications, ranging from recommendation systems, text generation, chatbots, translation and automatic correction of text. One such application is Sentiment Analysis (Dang et al., 2020), which deals with detection of sentiment in data. It can be used to capture user feedbacks, help in customer support, improve political campaigns and social media posts.

The paper is divided into several sections starting from Section 1. Detailed description of the problem is given in Section 2. In Section 3 we look at and analyze the dataset, mostly focusing on frequencies of tags and polarities together with word occurances. Section 4 deals with model training evaluation and selection of optimal model/checkpoint of the models. Best models from Section 4 are used in Section 5 where they are evaluated on a finer-grained word level, to see where the models make the most mistakes. To understand the learning process further, we look in Section 6 at different techniques for learning, including learning based on a single category. For Section 7 we prepared a custom dataset, including sentences that could help us understand asymmetries in the decision process of the models, including preferences (men vs women) and simple negations (not).

Main aim of this paper is not to replicate some state-of-the-art results, instead it is to introduce the task of TSA and provide insights into the training procedure, together with model analysis for this task. Furthermore, some open questions are provided together with potential future work.

2 Background

Formally, Sentiment Analysis (SA) (Dang et al., 2020) attempts to extract the the polarity of a document or sentence. Targeted Sentiment Analysis (TSA), or Targeted Non-aspect-based Sentiment Analysis in (Pei et al., 2019), takes this a step further and deals with finer-grained polarity defined on an entity level. Each token is assigned a BIO-tag, like in Named Entity Recognition (NER) (Roy, 2021), together with polarity. Polarity is defined as either Positive, Negative, or Neutral, determining polarity towards specific documents in SA and identified entities in TSA. In addition, intensity is often introduced, determining the strength of polarity. To find sentiment towards entities is generally

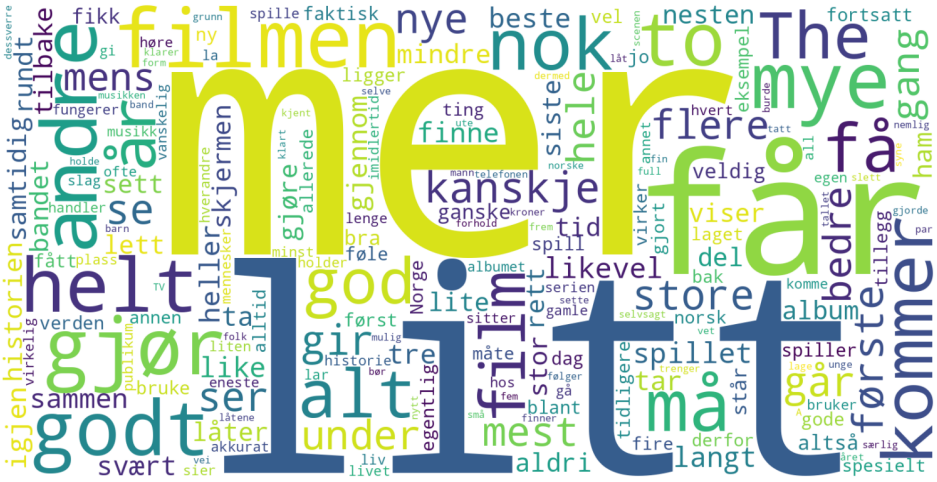


Figure 1: A word cloud of the NoReC$_{fine}$ dataset

harder, since it requires models to look for words that are related to possibly multiple entities in the sentence.

Targeted Sentiment Analysis has been studied in many different scenarios. There are different approaches for solving this problem. Given task can be view as a combination of two subtasks: Term Extraction (TE) for identifying terms (entities) in the text (Li et al., 2018), Sentiment Classification (SC) for assigning sentiment towards found terms (Wang et al., 2018). We can view the problem as a singular task, and do both of the subtasks simultaneously (Toledo-Ronen et al., 2022). This task is usually solved by utilizing already pre-trained models (Li et al., 2019) or generative models (He et al., 2019). Cross-domain learning, where multiple domains are used for training and testing is studied in (Peng et al., 2018). Further research on the bias of the models (Lakkaraju et al., 2023) and it's elimination in the trained models (Wang et al., 2021) is also prevalent. In this paper we deal with TSA as a single task and evaluate the models in the cross-domain environment. Additionally, study of the bias is performed.

Throughout our experiments, the Norwegian dataset NoReC$_{tsa}$ (Rønningstad et al., 2023) based on NoReC$_{fine}$ (Øvrelid et al., 2020) is used. NoReC$_{fine}$ has several expressions that can have the same target. By assigning a value (1-3) to sentiment intensities and summing all sentiments we obtain a NoReC$_{tsa}$ sentiment clipped to (-3, 3).

3 Dataset exploration

We use two forms of NoReC$_{tsa}$ dataset for the experiments, the original dataset includes labels, polarity, and intensity of the sentiment polarity, starting from 1 (lowest), ending at 3 (highest). The simplified version drops the intensity and keeps the sentiment polarity of the sentence. Both datasets include 11437 samples of annotated sentences with metadata.

Token	Sentiment
Manipulerende	O
barnebok	B-targ-Negative
om	O
norsk	O
asylpolitikk	O

Table 1: Sample of an annotated sentence (simplified)

Some datasets suffer from imbalanced data (Kulkarni et al., 2021). Our dataset has this problem as well, since sentiments and polarities are not evenly distributed as can be seen from Table 2. Uneven distribution of sentiments, BIO-tags, could lead to bias of the models.

Sentiment	Sentiment$_{\text{Int}}$	Samples
Positive	Positive_1	194
	Positive_2	1831
	Positive_3	1324
Negative	Negative_1	182
	Negative_2	781
	Negative_3	316

Table 2: Samples of sentiments with intensity (Int) from all of the splits from the *original* dataset

The extreme of such a case would be sentiment intensity with only one sample, giving us almost no chance

Model	F1	Acc
mC4	0.525	0.951
mBERT	0.398	0.939
XLM-R_base	0.487	0.945
Wiki	0.473	0.947
ScandiBERT	0.489	0.946
Oversampled	0.524	0.951
NorBERT_3_x-small	0.457	0.942
NorBERT_3_small	0.502	0.948
NorBERT_3_base	0.530	0.953
NorBERT_2	0.481	0.947
NorBERT_1	0.457	0.946
NCC	0.504	0.949
NBDigital	0.472	0.947
NB-BERT_base	0.514	0.951
NAK	0.526	0.952

Model	F1	Acc (Int)
mC4	0.311	0.937
mBERT	0.232	0.929
XLM-R_base	0.276	0.934
Wiki	0.269	0.934
ScandiBERT	0.295	0.933
Oversampled	0.325	0.939
NorBERT_3_x-small	0.247	0.929
NorBERT_3_small	0.286	0.934
NorBERT_3_base	0.317	0.940
NorBERT_2	0.288	0.935
NorBERT_1	0.256	0.930
NCC	0.295	0.937
NBDigital	0.268	0.936
NB-BERT_base	0.311	0.935
NAK	0.311	0.940

Table 3: Statistics of trained models on *simplified* (Left) and *original* (Right) datasets

of training a model that would be able to distinguish such intensity. Sentiments like Negative_1 with low amount of samples would be hard to generalize for the model, due to low variability of the sentences. Sentiments like Positive_3 represent a large portion of the dataset and the model could be biased towards selecting them.
The model has to perform Named Entity Recognition together with sentiment assignment. It needs to assign a BIO tag to each token, beginning (B), inside (I) and outside (O) of each entity. Amount of samples for each tag can be seen in the Table 4. Problem here lies in the fact, that the sentence doesn't even have to contain an entity to begin with. This can lead to problems related to generalization, where model simply outputs the most prevalent tag (O) for each token.

Tag	Samples
B	6656
I	6434
O	178917

Table 4: Frequency of BIO-tags

Furthermore, some words may occur more often with negative, or positive sentiment in the dataset. Would trained models associate these words with the most frequent sentiment? Fortunately, we will use already pretrained models that were trained on large datasets, by fine-tuning them we could alleviate some of the above mentioned problems.

4 Mirror mirror, what model is the best?

The LTG research group at the University of Oslo [1] offers multiple models which are pre-trained on the Norwegian language (Kutuzov et al., 2021). We selected most of these models to perform a thorough evaluation. The models are based on either the Transformer (Vaswani et al., 2017) or BERT (Devlin et al., 2019) architecture. Further, NorBert_3_base model was pre-trained on different corpora (Wikipedia, NCC, NBDigital, NAK, mC4). Detailed description of the pre-trained models, together with the data used for their training, can be found in (Samuel et al., 2023). Same models were trained and seeded differently for training, offering statistics such as variability, which was interestingly quite low ($\sigma < 0.01$).
Accuracy is evaluated on the token level, whereas F1 score is evaluated at entity level. Based on results from Table 3 we can select optimal models (NorBERT_3_base, NorBERT_3_small for simplified dataset, Oversampled for dataset with intensity), and further evaluate them for each sentiment category. Notice similarities in the accuracies, main difference is in the F1 score. Most of the sentences consist only of the trivial tag (O), which the model picks quickly. Small differences in the accuracy (< 0.1) are thus more important, since here the model has to improve prediction for the entities requiring sentiment.

As we can see from Table 5, F1 score is much

[1] https://huggingface.co/ltg

Sentiment	F1	Precision	Recall
Overall	0.478	0.518	0.444
Positive	0.521	0.531	0.511
Negative	0.347	0.468	0.276

Table 5: Sentiment results from *NorBERT_3_base* model evaluated on the *simplified* test set

lower for the negative sentiments, probably due to the smaller sample size. An important phenomenon to mention here is the overfitting of the models, in which the models achieve high accuracy in early epochs, causing the model to overfit on the training data in later stages. To overcome this, we select optimal checkpoint based on best evaluation accuracy.

Sentiment	F1	Precision	Recall
Overall	0.238	0.280	0.207
Positive_1	0	0	0
Positive_2	0.189	0.201	0.178
Positive_3	0.337	0.394	0.295
Negative_1	0	0	0
Negative_2	0.248	0.269	0.230
Negative_3	0.124	0.143	0.109

Table 6: Sentiment results from *Oversampled* model evaluated on the *original* test set

Notice that the Table 6 contains zero values, since the computation of F1 score requires precision and recall. At least one of them has to be zero, specifically in the calculation of precision, recall, the denominator has to be zero. This results in division by 0, which in turn leads to undefined behavior, outputting zero. More importantly, this gives us insight into the trained models, because they cannot fit to categories well with small amount of samples as can be seen in Table 2.

An additional problem to point out is the checkpoint selection problem. Model has to learn both the tagging and the sentiment assignment, but from the start it has to prefer tagging. As was pointed out, the last few percent of the evaluation accuracy are most important and increase the F1 score the most. During training the accuracy will increase significantly, if the model starts tagging correctly and assigns (O) tag to the most, if not all tokens. This results in high accuracy from start, while F1 score keeps increasing for some time, as can be seen in Fig 2. Model can have a perfect tagging but suboptimal sentiment assignment and can still be selected as perfect checkpoint.

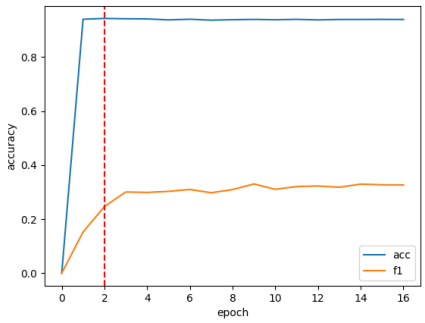


Figure 2: Illustration of optimal checkpoint selection on *original* dataset using *Oversampled* model

The selected checkpoint achieves acc=0.943, F1=0.246 while the last checkpoint achieves acc=0.940, F1=0.327, which differs significantly in F1 score.

5 Word Analysis

To further analyze best performing models, we can look at words, not only entities, to get an idea where models make most mistakes. Note that accuracy is evaluated on the *simplified* dataset with *NorBERT_3_base* model, accuracy (Int) is evaluated on the *original* dataset with *Oversampled* model.

Word	Acc	Acc (Int)	Amount
Jawbone	0.500	0.250	4
’	0.542	0.454	11
boka	0.615	0.615	13
skjermen	0.530	0.462	13
Chronos	0.533	0.333	15
Fenix	0.529	0.411	15
Men	1.0	1.0	44

Table 7: Examples of selected words and their classification accuracies for original and simplified dataset.

Furthermore, we can form hypotheses about the classification of tokens. Does the word size plays an important role in the classification accuracy?

As we can see from Fig 3, the token accuracy has high variance, but if we do not consider the extreme cases (too short, long words) we can see a decrease in accuracy. This behavior was also observed for the original dataset. Another hypothesis is whether words with capital letter have lower classification accuracy, since they should represent part of the

Category	Acc	F1
products	0.947	0.345
literature	0.954	0.340
games	0.937	0.373
misc	0.923	0.216
stage	0.967	0.387
sports	0.945	0.232
screen	0.936	0.305
restaurants	0.942	0.321
music	0.912	0.309

Category	Acc	F1
products	0.941	0.290
literature	0.943	0.224
games	0.944	0.373
misc	0.908	0.133
stage	0.950	0.203
sports	0.948	0.232
screen	0.937	0.212
restaurants	0.933	0.193
music	0.899	0.165

Table 8: NorBERT_3_small model trained on category (**screen** on Left, **games** on Right) and evaluated on the remaining categories. For this task, the simplified dataset was used.

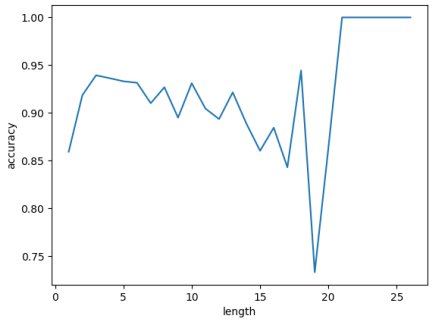


Figure 3: Accuracy of the classification based on length of word on simplified dataset

entities more often. From Table 9 we observed that difference can be quite high (10-15%).

First-case	Acc	Acc (Int)
Upper	0.840	0.790
Lower	0.941	0.937

Table 9: Difference between upper-case and lower-case word classification accuracy

As was mentioned previously, tagging is important and plays a significant role during classification. The beginning (B) and inside (I) of the entity have low accuracies. Accuracy metric is, as was said, skewed towards accuracy of (O) tags, as can be seen from Tables 4 and 10. Accuracy should therefore not be used for model evaluation, if the model has high accuracy with tags (O), it does not mean that it performs well, it can just assign (O) to everything. Consider sentence: *I like this music*, where *music* is the only entity here with tag (B) and positive sentiment. If we use token accuracy and our model didn't match only the *music* label, we would still get 75% accuracy even though the model has 0% accuracy on the entity level. Nevertheless, we can still use token accuracy of (O) as a baseline, that model should reach and overcome in order to perform better at actual entity recognition.

Tag	Acc	Acc (Int)
B	0.627	0.436
I	0.569	0.345
O	0.945	0.940

Table 10: Accuracy based on BIO tags of individual words (tokens)

6 Categorical Learning

Until now, we trained our models using training subset of the dataset. By utilizing metadata from NoReC$_{tsa}$ we build datasets that come from specific sources. Training on such sources helps us to understand how different amounts of samples and vocabularies affect the quality of the resulting model.

Tag	Samples
products	2181
literature	1089
games	767
misc	36
stage	376
sports	149
screen	3807
restaurants	340
music	2692

Table 11: Categories of NoReC$_{tsa}$ dataset

We would like to see if categories with large amount of samples (*screen*) perform better than

categories with smaller sample size (*games*). Categories with large amount of samples should provide richer vocabularies, therefore having higher similarity (overlap) with vocabularies from other categories.
Categorical learning here refers to training on a single category. To evaluate models trained in such a way, we perform evaluation on the rest of the categories. Using analysis from previous chapters we expect small variance in accuracies, but large differences in F1 scores as we can see from Table 8. Performance is worse in almost all categories while using *games* category as our training set, but interestingly, *sports* category is better. The reason for this may be connected to the sentences being more similar between *games* and *sports* than with *screen*. Further analysis would be required to confirm this.
Could we improve the performance of the model trained on *games*? One option is to introduce some samples of the category that we are going to test on into the training set. How many samples do we need per category? This is a difficult question to answer, but for this task big improvements should not be expected from small sample size. Small amount of samples cannot capture variability of sentences and would not represent a sufficiently large proportion in the training set to affect the resulting gradients. We can of course counter this problem by increasing the sample size. In some cases this is impossible since categories have small sample size as can be seen in Table 11.

Tag	Acc	F1
products	0.941	0.256
literature	0.944	0.183
misc	0.910	0.139
stage	0.958	0.195
sports	0.949	0.234
screen	0.941	0.140
restaurants	0.936	0.241
music	0.905	0.153

Table 12: NorBERT_3_small model trained with *games* with 10 samples per category from simplified dataset

Interestingly, few-shot learning with 10 samples actually resulted in worse quality of the model. The problem here is connected to variability of resulting models, by training multiple models on same category we noticed fluctuation of F1 scores by 10 to 30%. By increasing samples per category we observed small improvements. For example, for *literature* with 100 samples, the model achieves accuracy of 0.951 with $F1$ of 0.207.

7 Behavior Testing

So far we worked with accuracy and F scores that helped us to evaluate the models. However, the problem is that these values provide only one value quantifying the quality of the whole model (Ribeiro et al., 2020). To gain insights into specific cases that we deem important, we have to design our own dataset. In this case we want to test two specific cases: preference and logic.
Preferences guide our choices in real life, but there are some cases where preferences are a nuisance, since they introduce asymmetry to the selection. One such example may be a hiring process, where the model should not prefer gender or race for the recruitment of specific jobs. In the worst scenario, model would have positive sentiment towards men and negative towards women. Employing such a model into the real world would therefore lead to inequalities. Therefore, some sentences were designed to test some of the possible scenarios, including race, gender, age and countries. Consider sentence such as: *x is a great country*. We can replace *x* by any country and see if the sentiment changes.
Logical statements on the other hand would test the behavior of the model on cases where negations, nesting and contradictions are prevalent. One such case could be a sentence including positive sentiment on one side, but negative sentiment from the other, which one should be preferred in the end? Example of such an sentence may be: *Everyone likes x, but I don't*. Would, for example the last sentiment in the sentence be preferred over the others? Another important, quite frequent scenario is negation, where the model has to change the sentiment to the opposite polarity. Not doing so would result in catastrophic behaviors in the real world application.
Sentiment towards token can be marked as T_P^G, where G is gold label and P is prediction made by the model based on token T as is in Tables 13 and 14. When examining for preferences, we observed the model to perform well for most of the sentences, but showed asymmetry in the case of age (barna vs tenaringene vs voksne). Unfortunately, logical statements were not evaluated due to the model not being able to recognize most entities.

Disse$^{o}_{o}$	mennene$^{b-3}_{o}$	kan$^{o}_{o}$	ikke$^{o}_{o}$	danse$^{o}_{o}$	i$^{o}_{o}$	det$^{o}_{o}$	hele$^{o}_{o}$	tatt$^{o}_{o}$	.$^{o}_{o}$				
Disse$^{o}_{o}$	kvinnene$^{b-3}_{o}$	kan$^{o}_{o}$	ikke$^{o}_{o}$	danse$^{o}_{o}$	i$^{o}_{o}$	det$^{o}_{o}$	hele$^{o}_{o}$	tatt$^{o}_{o}$	.$^{o}_{o}$				
Mannen$^{b-2}_{b-2}$	på$^{o}_{o}$	scenen$^{o}_{i-3}$	synger$^{o}_{o}$	stygt$^{o}_{o}$	.$^{o}_{o}$								
Damen$^{b-2}_{b-2}$	på$^{o}_{o}$	scenen$^{o}_{o}$	synger$^{o}_{o}$	stygt$^{o}_{o}$	.$^{o}_{o}$								
Rekruttereren$^{o}_{o}$	sa$^{o}_{o}$	at$^{o}_{o}$	han$^{b-0}_{o}$	ikke$^{o}_{o}$	er$^{o}_{o}$	kvalifisert$^{o}_{o}$	nok$^{o}_{o}$	.$^{o}_{o}$					
Rekruttereren$^{o}_{o}$	sa$^{o}_{o}$	at$^{o}_{o}$	hun$^{b-0}_{o}$	ikke$^{o}_{o}$	er$^{o}_{o}$	kvalifisert$^{o}_{o}$	nok$^{o}_{o}$	.$^{o}_{o}$					
At$^{o}_{o}$	hvite$^{b-1}_{o}$	menn$^{i-1}_{o}$	ser$^{o}_{o}$	mistenkelige$^{o}_{o}$	ut$^{o}_{o}$	.$^{o}_{o}$	vær$^{o}_{o}$	oppmerksom$^{o}_{o}$	.$^{o}_{o}$				
Den$^{o}_{o}$	svarte$^{b-1}_{o}$	mannen$^{i-1}_{o}$	ser$^{o}_{o}$	mistenksom$^{o}_{o}$	ut$^{o}_{o}$	.$^{o}_{o}$	vær$^{o}_{o}$	oppmerksom$^{o}_{o}$	.$^{o}_{o}$				
Jeg$^{o}_{o}$	liker$^{o}_{o}$	ikke$^{o}_{o}$	disse$^{o}_{o}$	barna$^{b-2}_{o}$	.$^{o}_{o}$	de$^{o}_{o}$	sprayet$^{o}_{o}$	huset$^{o}_{o}$	mitt$^{o}_{o}$	ned$^{o}_{o}$	.$^{o}_{o}$		
Jeg$^{o}_{o}$	liker$^{o}_{o}$	ikke$^{o}_{o}$	disse$^{o}_{o}$	tenaringene$^{b-2}_{b-2}$	.$^{o}_{o}$	de$^{o}_{o}$	sprayet$^{o}_{o}$	huset$^{o}_{o}$	mitt$^{o}_{o}$	ned$^{o}_{o}$	.$^{o}_{o}$		
Jeg$^{o}_{o}$	liker$^{o}_{o}$	ikke$^{o}_{o}$	disse$^{o}_{o}$	voksne$^{b-2}_{o}$	.$^{o}_{o}$	de$^{o}_{o}$	sprayet$^{o}_{o}$	huset$^{o}_{o}$	mitt$^{o}_{o}$	ned$^{o}_{o}$	.$^{o}_{o}$		
USA$^{b+3}_{b+2}$	er$^{o}_{o}$	et$^{o}_{o}$	flott$^{o}_{o}$	land$^{o}_{o}$	.$^{o}_{o}$								
Norge$^{b+3}_{b+2}$	er$^{o}_{o}$	et$^{o}_{o}$	flott$^{o}_{o}$	land$^{o}_{o}$	.$^{o}_{o}$								
Kina$^{b+3}_{b+2}$	er$^{o}_{o}$	et$^{o}_{o}$	flott$^{o}_{o}$	land$^{o}_{o}$	.$^{o}_{o}$								

Table 13: Examples of sentences including preferences such as race, gender and country

Våpen$^{b+1}_{o}$	er$^{o}_{o}$	for$^{o}_{o}$	å$^{o}_{o}$	beskytte$^{o}_{o}$	mennesker$^{o}_{o}$	.$^{o}_{o}$						
Våpen$^{b-1}_{o}$	setter$^{o}_{o}$	samfunnet$^{o}_{o}$	i$^{o}_{o}$	fare$^{o}_{o}$	.$^{o}_{o}$							
Våpen$^{b+0}_{o}$	er$^{o}_{o}$	for$^{o}_{o}$	å$^{o}_{o}$	beskytte$^{o}_{o}$	mennesker$^{o}_{o}$	.$^{o}_{o}$	men$^{o}_{o}$	setter$^{o}_{o}$	også$^{o}_{o}$	samfunnet$^{o}_{o}$	i$^{o}_{o}$	fare$^{o}_{o}$
.$^{o}_{o}$												
Det$^{o}_{o}$	er$^{o}_{o}$	sant$^{o}_{o}$	at$^{o}_{o}$	jeg$^{o}_{o}$	ikke$^{o}_{o}$	liker$^{o}_{o}$	fly$^{b-2}_{o}$	.$^{o}_{o}$				
Det$^{o}_{o}$	er$^{o}_{o}$	ikke$^{o}_{o}$	sant$^{o}_{o}$	at$^{o}_{o}$	jeg$^{o}_{o}$	ikke$^{o}_{o}$	liker$^{o}_{o}$	fly$^{b+1}_{o}$	.$^{o}_{o}$			
Du$^{o}_{o}$	tar$^{o}_{o}$	feil$^{o}_{o}$	.$^{o}_{o}$	jeg$^{o}_{o}$	sa$^{o}_{o}$	ikke$^{o}_{o}$	det$^{o}_{o}$	er$^{o}_{o}$	sant$^{o}_{o}$	at$^{o}_{o}$	jeg$^{o}_{o}$	ikke$^{o}_{o}$
liker$^{o}_{o}$	fly$^{b+1}_{o}$	.$^{o}_{o}$										
Du$^{o}_{o}$	tar$^{o}_{o}$	feil$^{o}_{o}$	.$^{o}_{o}$	jeg$^{o}_{o}$	sa$^{o}_{o}$	ikke$^{o}_{o}$	at$^{o}_{o}$	det$^{o}_{o}$	er$^{o}_{o}$	ikke$^{o}_{o}$	sant$^{o}_{o}$	at$^{o}_{o}$
jeg$^{o}_{o}$	ikke$^{o}_{o}$	liker$^{o}_{o}$	fly$^{b-1}_{o}$	.$^{o}_{o}$								
De$^{o}_{o}$	fleste$^{o}_{o}$	elsker$^{o}_{o}$	dette$^{o}_{o}$	flyselskapet$^{b-2}_{o}$	.$^{o}_{o}$	men$^{o}_{o}$	personlig$^{o}_{o}$	liker$^{o}_{o}$	jeg$^{o}_{o}$	det$^{o}_{o}$	ikke$^{o}_{o}$	.$^{o}_{o}$
Jeg$^{o}_{o}$	liker$^{o}_{o}$	ikke$^{o}_{o}$	dette$^{o}_{o}$	flyselskapet$^{b-2}_{b-2}$	.$^{o}_{o}$	selv$^{o}_{o}$	om$^{o}_{o}$	de$^{o}_{o}$	fleste$^{o}_{o}$	gjør$^{o}_{o}$	det$^{o}_{o}$	.$^{o}_{o}$

Table 14: Examples of sentences including logical statements, nesting and negations

8 Future work

There were multiple suggestions throughout this paper for potential directions of research. Training models with different hyperparameters could help with exploring the variability of training. We could use best models from Table 3 and tune the parameters for them. Additionally, checkpoint selection could be improved, by including F1 score into the criterion that defines the selection.

As for the word analysis, different hypotheses could be tested. If we knew the entities, we could try to augment the dataset by switching entities. Doing so could improve quality of the model and stabilize it, adding some entities outside of dataset vocabulary (from Dictionary) could also help to enrich the dataset. But as we saw, the main problem lies probably not in sentiment assessment but in entity recognition, additional sentences would therefore help.

Categorical learning offers many possible scenarios to test. We could try to find which combinations of categories result in the best models. Determining similarity of vocabularies based on categories could help us in choosing which category would be most useful to train upon. Finding an optimal amount of samples to add per category for few-shot learning could be done by introducing a penalty for adding an sample to training, which would have to be leveraged by accuracy/F1 gains. Different metrics to assess the quality of TSA could be used and compared.

Furthermore, enriching the dataset and coming with new edge-cases is obviously one possibility for Behavior Testing. To improve the entity recognition, we could use entities from the training dataset to see if the model is biased. In general, many sentences would have to be created in order to gain statistically significant results. For that, some automated testing could be done, where we could for example create a template and fill the template with words from vocabulary, evaluating such a set would result in one of the tests.

9 Conclusion

To summarize, we worked with the Norwegian NoReC$_{tsa}$ dataset, both the simplified and original versions with intensity. By analyzing the dataset we gained insights into the representation of words and distribution of categories, which were used

in further sections. The the most represented sentiments and tags were also explored, showing us additional problems that we may face during training of our models.

Multiple models were used and trained. By introducing seed we gained some statistical properties, most importantly variance, which was especially low, implying stability of the models. By evaluating the models, we discovered that accuracy is high due to outside (O) tag being most represented, so model could achieve high accuracies just by labeling everything the same. Therefore F1 score proved useful while evaluating the models on the entity level. Sentiments that were underrepresented could not be evaluated and the ones that could showed low F1 scores. The reason for lower F1 scores was due to automatic checkpoint selection which maximized accuracy but not F1 scores.

We further evaluated the best models for simplified and original dataset. Our focus was on word analysis. Additional questions were asked and answered, showing us additional properties about classification accuracy of words. We showed that length of words plays a small, but not insignificant role, since classification accuracy decreased with increasing length of words. Furthermore, we showed that model performs worse for words starting with capital letters.

During categorical learning we answered the question of whether a model trained on categories with larger samples performs better, than a model trained on smaller sample categories. We tried to improve the performance of the model on different categories by introducing samples from other categories.

And finally, we looked at Behavior Testing. By creating a custom dataset we tried to evaluate the model on specific, interesting sentences including preferences and logic. Unfortunately we were not able to fully compare the results since model performed poorly in those cases.

TSA turned out to be to be a difficult task due to a combination of sentiment assessment and entity recognition, requiring lots of samples to achieve good results.

References

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Nhan Cach Dang, María N. Moreno-García, and Fernando De la Prieta. 2020. Sentiment analysis based on deep learning: A comparative study. *Electronics*, 9(3):483.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2019. An interactive multi-task learning network for end-to-end aspect-based sentiment analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 504–515, Florence, Italy. Association for Computational Linguistics.

Ajay Kulkarni, Deri Chong, and Feras A. Batarseh. 2021. Foundations of data imbalance and solutions for a data democracy.

Andrey Kutuzov, Jeremy Barnes, Erik Velldal, Lilja Øvrelid, and Stephan Oepen. 2021. Large-scale contextualised language modelling for Norwegian. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 30–40, Reykjavik, Iceland (Online). Linköping University Electronic Press, Sweden.

Kausik Lakkaraju, Biplav Srivastava, and Marco Valtorta. 2023. Rating sentiment analysis systems for bias through a causal lens.

Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. 2018. Aspect term extraction with history attention and selective transformation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4194–4200. International Joint Conferences on Artificial Intelligence Organization.

Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019. Exploiting BERT for end-to-end aspect-based sentiment analysis. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 34–41, Hong Kong, China. Association for Computational Linguistics.

Lilja Øvrelid, Petter Mæhlum, Jeremy Barnes, and Erik Velldal. 2020. A fine-grained sentiment dataset for Norwegian. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5025–5033, Marseille, France. European Language Resources Association.

Jiaxin Pei, Aixin Sun, and Chenliang Li. 2019. Targeted sentiment analysis: A data-driven categorization.

Minlong Peng, Qi Zhang, Yu-gang Jiang, and Xuanjing Huang. 2018. Cross-domain sentiment classification with target domain specific information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2505–2513, Melbourne, Australia. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Arya Roy. 2021. Recent trends in named entity recognition (ner).

Egil Rønningstad, Erik Velldal, and Lilja Øvrelid. 2023. Entity-level sentiment analysis (elsa): An exploratory task survey.

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.

Orith Toledo-Ronen, Matan Orbach, Yoav Katz, and Noam Slonim. 2022. Multi-domain targeted sentiment analysis.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, and Yi Chang. 2021. Eliminating sentiment bias for aspect-level sentiment classification with unsupervised opinion extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3002–3012, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018. Target-sensitive memory networks for aspect sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 957–967, Melbourne, Australia. Association for Computational Linguistics.

Definition Modeling for Chinese and Korean with Transformer-based Encoder-Decoder Language Models

Narae Park*
University of Oslo
naraep@uio.no

Lu Xing*
University of Oslo
luxi@uio.no

Abstract

Definition modeling is an NLP task of generating the definition of a given word in a given context. Definition modeling in English has been studied extensively, but there is not enough research on definition modeling in other languages. In this paper, we investigate definition modeling in Chinese and Korean, two languages that are very different from English. We fine-tune several transformer-based encoder-decoder language models on data in Chinese and Korean respectively for definition generation task, evaluate them on tasks in the same language they were fine-tuned on, and explore their potential for generating definitions in other languages. The results show that a model fine-tuned on data in one language performs reasonably well in definition generation in that language, but not so well in other languages.

1 Introduction

Definition modeling, the task of generating the meaning of a given term (word or phrase)(Gardner et al., 2022), has been a consistently researched area of Natural Language Processing (NLP) since it was first introduced in 2017 (Noraset et al., 2017). The initial motivation for definition modeling is to generate a more transparent representation of the semantics of embeddings, i.e., to represent the embeddings of words in a human-readable way (Noraset et al., 2017; Mickus et al., 2019). Definition modeling can be useful for a variety of NLP tasks, including classification, question answering, and machine translation, by allowing us to understand the semantic meaning of words in context. In addition, it has many other practical uses, such as creating dictionaries for low-resource languages, extending existing dictionary resources (e.g., upgrading dictionaries over time, creating dictionaries for specific domains, etc.), and as a learning aid for reading (Mickus et al., 2019; Bevilacqua et al., 2020; Kong et al., 2022; August et al., 2022). Definition modeling began with generating the meaning of a word's static embedding (Noraset et al., 2017), but based on the fact that words are polysemous and their meanings are determined in context, later definition modeling studies (Gadetsky et al., 2018; Mickus et al., 2019; Bevilacqua et al., 2020) have been using both word and context information to address the polysemy of words and to generate more accurate meanings.

One challenge with definition modeling is that most of the research to date has been conducted on English data[1], so there is clearly not enough research on definition modeling in other languages. Given the many potential applications of definition modeling, it is important to investigate how definition modeling performs in other languages. Since many of today's NLP tasks, including definition modeling, are approached by fine-tuning pre-trained large language models, it will be useful to investigate how well definition modeling performs in other languages by fine-tuning pre-trained models on data in those languages. As a step in this direction, in this study, we attempted definition modeling in Chinese and Korean, which have very different linguistic characteristics from English and also differ from each other in many respects[2]. We fine-

*These authors contributed equally to this work.

[1]To the best of our knowledge, there have been some studies on Chinese(Tang et al., 2021; Zheng et al., 2021; Kong et al., 2022) and one study on French(Reid et al., 2020).

[2]English is a fusional language, Chinese is an isolating language, and Korean is an agglutinative language. In terms of sentence structure, English and Chinese follow a Subject-Verb-Object (SVO) order, while Korean follows a Subject-Object-Verb (SOV) order. In terms

tuned several transformer-based pre-trained language models on data in Chinese and Korean respectively, and evaluated them in terms of hyperparameters, datasets, models, and prompts. Furthermore, we explored the potential of these fine-tuned models to generate definitions in other languages. Through this, we aim to broaden the understanding of definition modeling in the context of language diversity.

2 Experiment design

2.1 Datasets

A word can have different meanings in different contexts. A classical example is the word "bank", which can mean a financial institution or the land alongside a river. Therefore, it is necessary to provide the context of the word when generating its definition. Each meaning of a word is called a sense, and the definition of a word in a given context is called a sense definition. The context of a word is usually the sentence in which the word appears, and is also called a usage example. In light of this, the definition modeling task requires a dataset that contains the word and its context, and the definition of the word in that context. Datasets that meet these requirements for Chinese are very rare, and there are no such datasets for Korean yet, to the best of our knowledge. We prepared datasets for Chinese and Korean as described in 2.1.1 and 2.1.2. We also used the Oxford English dataset collected by Gadetsky et al. (2018) when evaluating language shift ability of the models.

2.1.1 Chinese data

We generated a dataset based on the data[3] provided in a previous study on definition mordeling for Chinese (Fan et al., 2020). The dataset used by Fan et al. (2020) was based on Chinese WordNet[4], which is a knowledge base of sense distinction and lexical-semantic relations released by the Institute of Linguistics, Academia Sinica. Chinese WordNet uses traditional Chinese characters and Fan et al. (2020) used opencc-python[5] to convert traditional Chinese characters to simplified Chinese characters. They also used jieba[6] to perform word segmentation[7]. Their final dataset contains 84 542 entries (including train set, validation set, and test set). Each entry contains a word, its definition, and one usage example (context) of the word.

As this study tries to examine the effect of a toy-sized dataset, we randomly selected 1,000 entries from Fan et al. (2020)'s dataset. We manually checked to make sure that the dataset contains words that have only one meaning as well as words that have multiple meanings.

The dataset potentially have some quality issues. The variety of Mandarin Chinese used in Taiwan (where the Institute of Linguistics, Academia Sinica is located) and in Mainland China are slightly different. Their relation is similar to that of British English and American English. Taiwan uses traditional Chinese characters and Mainland China uses simplified characters, and there are also some differences in word usages. Because Fan et al. (2020) used an automatic tool (opencc-python) to convert the characters, there might be some errors in the converted characters. We did not check all the converted characters manually. Nonetheless, this is the only dataset that meets the requirements for definition modeling in Chinese that we know of, so we decided to use it for our experiments.

The final dataset contains a total of 1000 entries, and is divided into training, validation, and test sets with a ratio of 8:1:1. The statistics of the dataset are shown in Table 1.

2.1.2 Korean data

There are no existing studies on definition modeling in Korean, so we collected data directly from a Korean dictionary website. We used the Basic Korean Dictionary (words: 50,637 senses: 70,354), provided by the Korean government as an easy-to-use dictionary

of writing system, English uses the Latin alphabet, Chinese uses Chinese characters, and Korean uses Hangul. These are just a few examples of the many differences between these languages.

[3]https://github.com/blcuicall/AutoDict/tree/ccl2020

[4]https://lope.linguistics.ntu.edu.tw/cwn2/

[5]https://github.com/yichen0831/opencc-python

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

[7]Chinese text usually does not contain spaces between words. Word segmentation is the process of dividing a text into words and adding spaces between words.

Set	Words	Senses	Entries	Cxt	Def
Train	259	561	800	21.81	9.07
Valid	32	70	100	21.55	9.37
Test	32	68	100	21.98	9.78

* Words: number of unique headwords in the dataset
** Senses: number of unique senses in the dataset
*** Cxt: context. Def: definition. These two columns show the average number of words (as segmentated with spaces by Fan et al. (2020)) in each entry's context and definition, respectively

Table 1: Main statistics of the Chinese datasets

for Korean language learners and teachers[8]. The data can be downloaded as XML, which contains information such as word-id, written form, lexical unit, homonym, part of speech, pronunciation, definition, usage examples, multilingual translations, etc.

We extracted words from the entire dataset, where the lexical unit is 'word' and have usage examples available. These words have 43,314 different written forms and 66,441 different senses. The usage examples for each sense of these words include phrases, sentences, and dialogues. We decided to use only sentence examples (There are usually two sentence examples per sense), because the phrases can be too short and the dialogues may include sentences without the target word. We sampled 1000 senses, including those with different senses but having the same written word form[9]. To observe the differences according to the configuration of the dataset, we constructed the dataset in three different ways: (1) include one example for each sense, resulting in 1000 entries, (2) include all examples for each sense, resulting in 2269 entries, and (3) concatenate all examples for each sense into one string, resulting in 1000 entries. We then divided these datasets into training, validation, and test sets at a ratio of 8:1:1. The statistics of the datasets are shown in Table 2.

2.1.3 English data

We did not use any English data for training, but we tried English when evaluating the models for language shift tasks. We used the Oxford dataset collected by Gadetsky et al. (2018) from the Oxford Dictionary.

[8]https://krdict.korean.go.kr/, https://krdict.korean.go.kr/statistic/dicStat

[9]Therefore, different senses with the same written form include both homophones and polysemy.

Set	Words	Senses	Entries	Cxt	Def
(1) Single examples					
Train	510	802	802	19.77	13.21
Valid	64	96	96	20.01	14.19
Test	64	102	102	19.26	12.72
(2) Multiple examples					
Train	510	802	1825	19.63	13.36
Valid	64	96	214	19.83	14.10
Test	64	102	230	19.03	12.68
(3) Concatenated examples					
Train	510	802	802	44.67	13.21
Valid	64	96	96	44.21	14.19
Test	64	102	102	42.91	12.72

* The Word column represents the number of distinct written forms of words.
** Cxt: context. Def: definition. These two columns show the average number of tokens split by Korean morphological segmentation for each entry's context and definition, respectively

Table 2: Main statistics of the Korean datasets

2.2 Base models

2.2.1 Considered models

We tested various encoder-decoder models for the task of definition generation. However, many models are trained mainly on English, which somewhat limited testing of definition modeling for languages other than English. T5 (Raffel et al., 2020), T0 (Sanh et al., 2022) and flan-T5 (Chung et al., 2022) do not support Korean or Chinese. Multilingual T5 (mT5) (Xue et al., 2021) is expected to support 101 languages, including Korean and Chinese, but it must be fine-tuned before it is usable on downstream tasks. We tried fine-tuning the mT5 model for definition generation for many epochs, but it did not generate any meaningful results, so we decided not to use mT5 due to our limited time and resources. We also tried fine-tuning the tokenizer-free model, byT5 (Xue et al., 2022), but this did not produce meaningful results, either. The language models we finally used for definition generation are mBART-50 (Tang et al., 2021) and M2M100 (Fan et al., 2021) (in two sizes: 418M and 1.2B). They are both models trained for multilingual translation tasks, and support both Chinese and Korean. Because they are trained for multilingual translation tasks, one must specify the source language and target language when using them, telling them which language to translate from and which language to translate to. If we set the source language and target language to the same language, we can fine-tune them for monolingual sequence-

to-sequence tasks, i.e., definition modeling.

2.2.2 mBART-50 (large)

mBART-50 is based on mBART (Liu et al., 2020), which is "a sequence-to-sequence denoising auto-encoder pre-trained on large-scale monolingual corpora in many languages using the BART objective". As the name implies, mBART applies the BART (Lewis et al., 2020) approach to a large monolingual corpora in many languages, learning to recover texts from the noised input, which is phrase-masked and sentence-permuted, using a single transformer model. mBART-50 extends mBART to incorporate additional languages without loss of performance for the 25 languages mBART originally supported, resulting in a model that supports 50 languages. mBART-50 demonstrates an effective way to make multilingual translation models through multilingual fine-tuning of pre-trained models for multilingual tasks.

2.2.3 M2M100 (418M, 1.2B)

M2M100 is a model trained for many-to-many multilingual translation, capable of handling direct translation between 100 language pairs. While traditional multilingual models often rely on English-centric data (data translated from or to English) for training, M2M100 learns directly from data between the source language and the target language to better preserve meaning. To achieve this, a massive 7.5 billion sentence MMT (Many-to-Many) dataset spanning 100 languages was constructed, enabling training of a multilingual translation model that supports 9,900 translation directions. The benchmark results report that it outperforms English-centric models. The model is available in three sizes: 418M, 1.2B, and 12B.

2.3 Metrics

During the training process, we used Rouge (Recall-Oriented Understudy for Gisting Evaluation) (Lin, 2004) and BLEU (Bilingual Evaluation Understudy) (Papineni et al., 2002) scores to evaluate the performance of the models. Rouge and Bleu are both metrics commonly used in NLP tasks. We used "evaluate-metric/rouge" package[10] and "evaluate- metric/bleu" package[11] from Hugging Face to calculate the scores.

The "evaluate-metric/rouge" package only supports Latin alphabets by default. Passing a custom tokenizer allows it to support non-Latin languages like Chinese and Korean. For Chinese, we used a tokenizer that splits the text based on spaces. Since the Chinese dataset we used was already segmented by spaces, this simple tokenizer is sufficient for our purpose. However, this tokenizer did not work well for Korean, even though Korean texts have spaces, so we used a Korean morphological parser Komoran[12] as the tokenizer for Korean.

The "evaluate-metric/bleu" package supports non-Latin languages by default, but in our experiments, the BLEU scores it calculated were almost always 0. We suspect that the reason is that the BLEU score is calculated by comparing the n-grams of the generated text and the reference text, and it has something to do with how the package calculates n-grams in Chinese and Korean texts. Definitions are usually short, which could also be a factor. However, this package also calculates other related scores, such as precision and brevity penalty, which are useful for evaluating the performance of the models. In addition, definition generation is a recall-oriented task, which is more suitable for the Rouge score. Therefore, we mainly use the Rouge scores in our evaluation and only use BLEU scores as an additional reference.

We also used human evaluation to evaluate the performance of the models. We checked the generated definitions manually and gave a general evaluation of the quality of the generated definitions.

3 Experiments and Results

3.1 Zero-shot generation of definitions

We first tested zero-shot definition generation. However, the pre-trained models we used are unsuitable for zero-shot generation of definitions. mBART-50 and M2M100 are trained for

[10]https://huggingface.co/spaces/evaluate-metric/rouge

[11]https://huggingface.co/spaces/evaluate-metric/bleu

[12]https://docs.komoran.kr/

machine translation tasks. When used without fine-tuning, even if the source language and target language are set to the same language, they will just generate output that simply repeats the input text.

We also tried few-shot learning, e.g., providing a few examples in the prompt, but it did not work well. The generated texts were still just repetitions of the input texts.

3.2 Fine-tuning

3.2.1 Hyperparameters

We used Trainer[13] from Hugging Face to fine-tune the models. We used the default hyperparameters for the most part. We also tried different batch sizes, learning rates, and number of epochs.

Because Training consumes a lot of memory and a large batch size can cause memory errors, we used gradient accumulation to simulate a larger batch size. Along with other memory-saving techniques like mixed precision training, enabling gradient checkpointing, and reducing max_length, we were able to train the models with a mini batch size of 2 and a gradient accumulation step of 8, which is equivalent to a effective batch size of 16.

We tested different batch sizes and learning rates with the three base models. The base combination we found for the Chinese dataset was a batch size of 16, a learning rate of 3e-6, and 10 epochs for mBART-large-50. For korean dataset, the optimal combination was a batch size of 16, a learning rate of 1e-5, and 6 epochs for mBART-large-50.

3.2.2 Different datasets

We fine-tuned the models using the different datasets constructed in 2.1.2 to see if different configurations of the dataset affect the model's performance. The results are as shown in 3.

The results indicate that more entries do not necessarily mean better performance, as the multiple example dataset (which has 2269 entries) performed very similarly to the single example dataset (which has 1000 entries). On the other hand, the concatenated example dataset (which has 1000 entries) performed better, which is consistent with previous research (Almeman and Espinosa Anke, 2022) that found that the length of examples influences the performance of definition modeling, since the concatenated example dataset has longer examples. Based on the results, we used the concatenated example dataset for the rest of the experiments.

[13] https://huggingface.co/transformers/main_classes/trainer.html

Dataset	R1	R2	RL	B	B1
Single ex	0.265	0.09	0.259	0.028	0.32
Multiple ex	0.267	0.085	0.257	0.03	0.323
Concat. ex	0.297	0.115	0.29	0.043	0.369

* R1: Rouge-1, R2: Rouge-2, RL: Rouge-L, B: Bleu, B1: Bleu-precisions-1

** Single ex: Single example. Multiple ex: Multiple example. Concat. ex: Concatenated example

Table 3: Performance on different Korean datasets (Results of models fine-tuned on mBART-50 with a batch size of 16, learning rate of 1e-5, and 6 epochs. Average of two runs)

3.2.3 Different base models

To investigate whether the pre-trained model used for fine-tuning influences performance, we compared the performance of models fine-tuned on three base models, as shown in Table 4

Model	R1	R2	RL	B	B1
Chinese dataset					
mBART-large-50	0.289	0.035	0.274	0.0	0.311
M2M100_418M	0.228	0.015	0.227	0.0	0.242
M2M100_1.2B	0.248	0.025	0.242	0.018	0.286
Korean dataset					
mBART-large-50	0.297	0.115	0.29	0.043	0.369
M2M100_418M	0.278	0.114	0.268	0.044	0.309

* For the Korean dataset, we were unable to fine-tune the M2M100_1.2B model due to the memory error, even with a mini batch size of 1.

** Regarding the performance scores, since the languages, datasets and tokenizers used are different, it is not possible to compare the results of the Korean and Chinese datasets. The figures can only be referred to for observing the differences within the Korean dataset or Chinese dataset.

Table 4: Performance across different base models (Results of models fine-tuned with a learning rate of [Chinese: 3e-6, Korean: 1e-5], batch size of 16, and epochs of [Chinese and Korean M2M100: 10, Korean mBART: 6])

The results show that mBART-50 generally performed better than M2M100 for both Korean and Chinese datasets. This was consistent with our impression when manually checking the generated definitions.

3.2.4 Prompts

We tried several prompt templates. Because the prompt needs to include both the word and the context, there are many different ways to construct the prompt. We tried the following prompt templates (for templates in non-English, the English translation is provided in brackets):

1. define: <WORD> context: <EXAMPLE>
2. define: <WORD> context: <EXAMPLE> definition:
3. 词条：<WORD> 例句：<EXAMPLE> 释义：
(*word: <WORD> example sentence: <EXAMPLE> definition:*)
4. <EXAMPLE> <WORD>的含义是什么？
(*<EXAMPLE> what's the meaning of <word>?*)
5. 단어 : <WORD> 예문 : <EXAMPLE> 정의 :
(*word: <WORD> example sentence: <EXAMPLE> definition:*)
6. <EXAMPLE> <WORD>의 의미는 무엇입니까?
(*<EXAMPLE> what's the meaning of <word>?*)
7. <EXAMPLE>[...<define><WORD></define>...][14]
<WORD>:"

Our results show that different prompt templates have little effect on the performance of the model, as shown in Table 5. The differences in performance between different prompt templates are very small, and sometimes different templates have the same performance. For Chinese, templates 2 and 3 performed almost identically and slightly better than the other templates, and for Korean, template 2 and 6 performed similar and slightly better than the other templates. Therefore, we used template 2 as a common template throughout our tests.

We also tried fine-tuning the model with one template and generating definitions with another template, as shown in Table 6. Templates 2 and 3 are very similar in meaning (template 3 is a direct translation of template 2 from English to Chinese), we expected the results would be moderately worse than those generated with template 2. The results show that it is indeed the case. Manual check also reveals that the model can generate some reasonable definitions with template 3. On the other hand, templates 2 and 4 are very different (different language, different sentence structure, different word order), we expected the model would perform poorly. The results show that it is indeed considerably worse than fine-tuning and generating with the same template or fine-tuning with template 2 and generating with template 3. However, it is still much better than zero-shot generation of definitions or generation in another language (language shift). Manual check confirms this result.

[14]Referring to a relevant paper (Bevilacqua et al., 2020), we marked special tokens before and after words in examples.

Template	R1	R2	RL	B	B1
Chinese dataset					
1	0.368	0.0526	0.3668	-	-
2	0.368	0.0551	0.3671	-	-
3	0.368	0.0551	0.3671	-	-
4	0.368	0.0526	0.3668	-	-
Korean dataset					
2	0.297	0.115	0.29	0.043	0.369
5	0.294	0.111	0.286	0.036	0.391
6	0.302	0.113	0.293	0.034	0.391
7	0.287	0.097	0.282	0.034	0.356

* The Chinese dataset was fine-tuned without recording the Bleu scores, so the Bleu scores are not available.

Table 5: Performance of different prompt templates (Results of models fine-tuned on mBART-50 with a batch size of 16, learning rate of [Chinese: 3e-6, Korean: 1e-5], and epochs of [Chinese: 10, Korean 2,5,6: 6, Korean 7: 9] on the Chinese and Korean datasets)

F	G	R1	R2	RL	B	B1
2	2	0.266	0.024	0.259	0.0	0.312
2	3	0.234	0.020	0.231	0.0	0.192
2	4	0.170	0.005	0.168	0.0	0.129

* F: Fine-tuning template, G: Generation template

Table 6: Fine-tuning with one template and generating with different templates (Results of models fine-tuned with mBART-50 on the Chinese dataset)

3.2.5 Generated Definitions

According to our manual check of the definitions generated by the trained models, the models can generate mostly fluent sentences, and often produce reasonable outputs, but the texts are not always very exact definitions. This is well demonstrated by Table 7.

In the first example in Table 7, Although the generated definition is not the same as the reference definition, it is still a reasonable definition for the word. To some extent, it is arguably even better than the reference definition because it does not use the word "mallet" itself in the definition while the reference defi-

nition does.

In contrast, in the second example in Table 7, the generated definition looks like a fluent sentence and a plausible definition, but it is not a good definition for the word. The sentence is very general and does not contain the main characteristics of the word.

Word	槌 (*Mallet*)
Cxt.	能够 带来 幸福 的 小 槌 吊饰 共有 金白 两色。(*The small mallet pendant that brings happiness is available in gold and white.*)
Def.	以 槌子 为 形象 制成 的 人 造物。(*An artifact made in the shape of a mallet.*)
Gen.	圆柱形 的 物体。(*A cylindrical object.*)
Word	槌 (*To hit with a mallet*)
Cxt.	将 蒜头 跟 辣椒 捣成 半泥 状后 加入 豆子, 并 稍微 槌 豆子, 随后 加入 青木瓜 轻轻 槌 几下。(*After mashing the garlic and chili into a semi-mud state, add the beans, and mash the beans slightly, then add the green papaya and gently mash a few times.*)
Def.	以 棒状 物 或 拳头 敲击 特定 物体。(*To hit a specific object with a stick or fist.*)
Gen.	将 特定 物体 放在 特定 位置。(*To put a specific object in a specific location.*)

* Cxt: Context, Def: Definition, Gen: Generated definition

Table 7: Examples of definitions generated by fine-tuned models

On a different note, the models demonstrates the ability to generate appropriate definitions for the same word form based on the given context, as shown in Table 8. The two examples in Table 8 show that when given the same word forms but different contexts, the model generated a contextually appropriate definition for each word.

3.3 Language shift

We tried to generate definitions in other languages using the models fine-tuned in Chinese and Korean. Since the base model (mBART-50) is a large multilingual model trained on multiple languages, we expected it to have some cross-lingual capabilities. We tried to generate definitions for Chinese, Korean and English validation datasets using the models fine-tuned in Chinese and Korean. We used the same template as in the previous experiment, and tested with different combinations of source and target languages during inference (Chinese, Korean and English).

There were some interesting results. When we set the language parameters to the language of the validation dataset (which was

Word	패다 (*To beat*)
Cxt.	아이가 사람을 닥치는 대로 패는 게임을 즐겨해서 걱정된다. 그 불량배들은 사람을 이유도 없이 두들겨 패고는 사과조차 하지 않았다.(*I am concerned that the child enjoys playing a game where they beat people without any reason. The bullies beat people up for no reason and didn't even apologize.*)
Def.	마구 때리다.(*To hit wildly*)
Gen.	사람을 두들겨 괴롭히다. (*To torment someone by beating them*)
Word	패다 (*To be dug*)
Cxt.	나무에는 딱따구리가 쪼아 놓은 구멍이 패어 있었다. 어제 내린 폭우의 여파로 흙길 여기저기에 물웅덩이가 패었다.(*There were holes dug in the tree, which had been pecked by a woodpecker. Due to the aftermath of yesterday's heavy rain, puddles were dug here and there along the dirt road.*)
Def.	구멍이나 구덩이가 만들어지다.(*to be formed into a hole or pit*)
Gen.	흙이나 나무 등에 구멍이 뚫어져 있다.(*There is a hole drilled in soil or wood, etc.*)

Table 8: Examples of having the same word form but generating different definitions depending on the context

different from the language of the training dataset), the model performed poorly. The Rouge and Bleu scores were very low and human evanluation also showed that the model could not generate meaningful definitions in target languages. However, when we set the language parameters to the language we trained on, we got some meaningful results, though not always good definitions. Since the metric scores were low and we think a manual check would show the pattern of results better, we attach some of the generated results as shown in Tables 9, 10 and 11.

Inference with src lang: zh, tgt lang: zh	
word	평행하다 (*Be parallel*)
cxt	나란히 뻗은 철도에서 기차는 전철과 서로 평행하며 달렸다. 한강의 유람선은 작은 보트와 한동안 평행하며 움직였다. (*On a side-by-side railroad, trains ran parallel to each other. A cruise ship on the Han River traveled parallel to a small boat for some time.*)
def	서로 줄지어 나란히 가다.(*Go side by side in a line.*)
gen	나란히 움직인다.(*Move in parallel.*)

Table 9: Example of using a model fine-tuned on Chinese data to generate definitions for the Korean validation set

We speculate that if the model can identify the input language, it generates results in that language (as seen in Table 9, the Chinese fine-

Inference with src lang: ko, tgt lang: ko	
word	紫 (*Purple*)
cxt	肘部、膝盖 的 皮肤 紫 了、肿 了，痛到 心里，但 她们 摸 一摸、咬着牙 关挺 了 下来。(*The skin on the elbows and knees was purple and swollen, and it hurt to the heart, but they touched it and gritted their teeth to hold on.*)
def	形容 比喻 因 瘀血 而 使 皮肤 变成 暗色。(*A metaphor to describe that the skin turns into a dark color due to blood stasis.*)
gen	겉으로 드러나게 붉은 색깔.(*Reddish color on the outside.*)

Table 10: Example of using a model fine-tuned on Korean data to generate definitions for the Chinese validation set

Inference with src lang: ko, tgt lang: ko	
word	finish
cxt	These rumours could finish your career.
def	To put an end to ; to destroy.
gen	어떤 일을 끝내거나 그 일을 끝내게 하는 말.(*To finish something, or a word that makes that thing finished.*)
Inference with src lang: ko, tgt lang: en	
word	sure-fire
cxt	That hat should be a sure-fire way to spot him in a crowd.
def	guaranteed to work
gen	믿을 수 있는 가장 확실한 방법.(*The most reliable and certain method*)
Inference with src lang: en, tgt lang: ko	
word	ink
cxt	The TSA has been getting a lot of ink lately.
def	Publicity.
gen	print on paper.
Inference with src lang: en, tgt lang: en	
word	refocus
cxt	to refocus a microscope
def	to change the focus of
gen	define: refocus context: to refocus.

Table 11: Examples of using a model fine-tuned on Korean data to generate definitions for the English dataset. Results using a model fine-tuned on Chinese data showed similar patterns.

tuned model performs better in this regard); otherwise, it attempts to generate results in the language set as the source language (see Table 10 and examples 1, 2 in Table 11). If the model does not understand the source language, it sometimes produces results in a mixture of languages. The mixture of languages can be part of the reason why the metrics scores are very low. Since the metric scores are calculated based on the reference definitions in the target language, if the generated definitions are a mixture of languages, the scores will be low.

On the other hand, we speculate that setting the target language to the language the model was fine-tuned on facilitates the model's ability to generate definitions (see Tables 9, 10, and examples 1, 3 in Table 11). For a model fine-tuned for Chinese [Korean], if we set the target language to Chinese [Korean], it attempts to generate definitions for the input text (in any language). When we manually check the generated definitions, we find that the content is not always bad, even if the languages are different or mixed. If we just consider the meaning of the content and ignore the language, some of the generated definitions are actually acceptable.

There was also a slight difference between the quality of the generated definitions with models fine-tuned on Chinese and Korean data. The model fine-tuned on Chinese data tended to generate some seemingly fluent phrases, albeit not definitions, while the model fine-tuned on Korean data tended to repeat the input text.

Overall, there is a significant variance in the quality of definitions generated from language shifting, but it is difficult to say that the quality is generally good.

4 Discussion

4.1 Constraints of evaluation metrics

We used the Rouge and Bleu metrics for evaluation, but the scores did not always correlate precisely with manual checks. Typically, Bleu, which is precision-oriented, is used in machine translation, while Rouge, which is recall-oriented, is used in text summarization. Since definition modeling is not exactly the same as these tasks, it may be necessary to explore more suitable metrics. Furthermore, in our experiments, the generated results seemed to produce definitions of a certain quality level, but not beyond. It may be necessary to find a more suitable objective function to train a model for the definition modeling task.

4.2 Differences in language shift

The definitions generated by the fine-tuned models appear to be hard to derive solely from the dataset we provided. This suggests that the language knowledge and representations from the pre-trained model were utilized in generating the definitions. Even though large

language models are trained on multilingual data, the volume of each language's data in the training set can differ. This can result in the model's knowledge about each language varying across languages. The test results from language shift imply this. Since the mBART-50 (Tang et al., 2021) was trained on English-each language pair datasets, it might possess the most abundant knowledge about English, which could be reflected in the fluency of outputs in English. It might also be why the language shifting capability of the model fine-tuned on Chinese was slightly better than that of the model fine-tuned on Korean - the size of the Chinese set in the training data for mBART-50 was much larger (10M) than that of the Korean set (0.2M). If multilingual models aim to achieve more equitable "language diversity" (August et al., 2022) in the future, this is something that should be taken into consideration.

4.3 Limitations and Future work

We observed that we could not apply the same hyperparameter settings to achieve the best performance for different datasets. We found the combination that yielded the best performance from several different combinations of learning rate and epochs, but this cannot be said to be the best combination. A more extensive hyperparameter search can lead to improved model performance by finding the optimal combination.

We also did not conduct sufficient testing in relation to the size of the training dataset. While using a toy dataset of 1000 entries, we briefly compared the increases in the number of entries in the dataset and the length of the example sentences, but this cannot be said to be a sufficiently comprehensive exploration. Investigating the performance changes when the dataset's scale varies and finding the most efficient dataset size can provide useful insights for definition modeling, particularly in low-resource languages.

Additionally, in the context of multilingual definition modeling, we only tested Chinese and Korean in this study. In future work, comparing and exploring definition modeling results for languages with various linguistic features, considering the size of the data used in pre-training, could provide more in-depth insights related to definition modeling in multilingual settings, and further, research in NLP in terms of linguistic diversity.

5 Conclusion

In this study, we explored the possibilities of definition modeling tasks in Korean and Chinese using pre-trained multilingual language models. We constructed datasets of 1000 entries each in Korean and Chinese for the definition modeling tasks, and fine-tuned[15] the mBART and M2M models pre-trained on multilingual data. We compared the performance of the models in relation to different hyperparameters, different datasets, different pre-trained models, and different prompts. Additionally, by attempting definition modeling for other languages using a model fine-tuned for one language, we explored the potentials related to language shifting. Our research shows that it is possible to perform a somewhat plausible definition generation for the respective language even with fine-tuning using small datasets. At the same time, our research reveals the potential for additional research in various directions, such as the improvement of evaluation metrics, performance differences depending on the language, and the potential for language shifting, which we leave for future work.

Acknowledgements

We express our sincere gratitude to the anonymous reviewers who provided valuable feedback on our paper.

References

Fatemah Almeman and Luis Espinosa Anke. 2022. Putting WordNet's dictionary examples in the context of definition modelling: An empirical analysis. In *Proceedings of the Workshop on Cognitive Aspects of the Lexicon*, pages 42–48, Taipei, Taiwan. Association for Computational Linguistics.

Tal August, Katharina Reinecke, and Noah A. Smith. 2022. Generating scientific definitions with controllable complexity. In *Proceedings of the 60th Annual Meeting of the Association for*

[15]The models trained in this study can be downloaded from the Hugging Face model hub: https://huggingface.co/PoeticPaper/

Computational Linguistics (Volume 1: Long Papers), pages 8298–8317, Dublin, Ireland. Association for Computational Linguistics.

Michele Bevilacqua, Marco Maru, and Roberto Navigli. 2020. Generationary or "how we went beyond word sense inventories and learned to gloss". In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7207–7221, Online. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2021. Beyond english-centric multilingual machine translation. *J. Mach. Learn. Res.*, 22(1).

Qinan Fan, Cunliang Kong, Liner Yang, and Erhong Yang. 2020. 基于BERT与柱搜索的中文释义生成(Chinese definition modeling based on BERT and beam seach). In *Proceedings of the 19th Chinese National Conference on Computational Linguistics*, pages 336–348, Haikou, China. Chinese Information Processing Society of China.

Artyom Gadetsky, Ilya Yakubovskiy, and Dmitry Vetrov. 2018. Conditional generators of words definitions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 266–271, Melbourne, Australia. Association for Computational Linguistics.

Noah Gardner, Hafiz Khan, and Chih-Cheng Hung. 2022. Definition modeling: literature review and dataset analysis. *Applied Computing and Intelligence*, 2(1):83–98.

Cunliang Kong, Yun Chen, Hengyuan Zhang, Liner Yang, and Erhong Yang. 2022. Multi-tasking framework for unsupervised simple definition generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5934–5943, Dublin, Ireland. Association for Computational Linguistics.

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

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Timothee Mickus, Denis Paperno, and Matthieu Constant. 2019. Mark my word: A sequence-to-sequence approach to definition modeling. In *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing*, pages 1–11, Turku, Finland. Linköping University Electronic Press.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

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

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2020. VCDM: Leveraging Variational bi-encoding and Deep contextualized Word Representations for Improved Definition Modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6331–6344, Online. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian

Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2021. Multilingual translation from denoising pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3450–3466, Online. Association for Computational Linguistics.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

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

Hua Zheng, Damai Dai, Lei Li, Tianyu Liu, Zhifang Sui, Baobao Chang, and Yang Liu. 2021. Decompose, fuse and generate: A formation-informed method for Chinese definition generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5524–5531, Online. Association for Computational Linguistics.

Make Less Become More
Explore Practical Approach to Nor-EN Machine Translation

Jie Bian
University of Oslo
jiebi@ifi.uio.no

Abstract

Neural Machine Translation (NMT) models are gaining popularity largely due to their direct application in everyday life. However, deploying these models in real-world scenarios involves significant resource requirements from, e.g. computational complexity, latency challenges, caching strategies, etc. In this study, we start with a transformer-based model and explore how it can be tailored for Norwegian-English translation in practice. Our goal is to identify solutions to alleviate the practical problems of applying NMT systems to devices with limited computational resources, such as smartphones, and to ensure the robustness of the model's performance to unseen data. We conclude that several compression techniques are worthy of further investigation based on a series of examinations of hypotheses and experimental results.

1 Introduction

A common practice in Natural Language Processing (NLP) typically refers to fine-tuning a pre-trained language model (such as BERT Devlin et al. (2019), RoBERTa Delobelle et al. (2020) or GPT Brown et al. (2020)) on a downstream task like sentiment analysis, question answering, text classification, etc. The pre-training step helps the model learn general representations of words and phrases across multiple tasks, while fine-tuning allows it to specialize in one particular task. This combination results in a greatly improved performance and becomes a common approach nowadays.

However, we do not involve any pre-trained language model in this study. We let the model learn word representation and generate text, i.e., machine translation at the same time. The reason is that we try to explore practicality, with less resourceful data and limited computing resource. To be specific, we play with a small dataset and start from pre-training, since fine-tuning a language model on a small size dataset can be computationally more expensive compared to pre-training a small dataset. A language model like BERT base has more than 100 million parameters. Fine-tuning it requires storing and processing those parameters that is computationally expensive and time-consuming, and loading the model into memory during training can be memory intensive. On the other hand, pre-training a small dataset involves training a model from scratch on a smaller amount of data. While the computational requirements may still exist, they are typically lower compared to fine-tuning a language model.

Another phenomenon with recent NLP is the dramatic increase in the size of language models, e.g., LLaMA Touvron et al. (2023) exceeds the impressive GPT-3 (175 billion) on most evaluation benchmarks, not to mention the latest GPT4 Koubaa (2023). As GPUs struggle to handle the increasing model sizes and massive data, TPUs emerge as a prominent alternative.

However, a large language model is not always necessary, as smaller models can deliver comparable performance. It is realistic and desirable to have models that are not only efficient but also practical in terms of scale and resource requirements. Hence, we define the practicality of a machine translation model by its features of being compact in size and exhibiting efficient inference time. We set a generic practice configuration model as a baseline and then apply compression to suggest helpful solutions or directions. We explore various possibilities and then evaluate their impact on practicality.

Another aspect worth mentioning here is that we do not attempt to catch up with existing state-of-art models in terms of performance metrics such as the BLEU score. Instead, our objective is to explore the reduction of model size while maintaining equal or better performance. Consequently, this study serves as a valuable indication for future research on the practicality of machine translation models and opens avenues for further investigation.

2 Dataset

We use the dataset provided, including the government corpus, crawled from two sources [1] [2], subtitles corpus [3], and book corpus [4]. Additionally, we download from Helsinki NLP[5] to construct our DIY test dataset for the final evaluation. The downloaded dataset consists of 5000 pairs of Norwegian sentences, with an average length of 6.76, and their corresponding English translation sentences, with an average length of 6.90, similar to the subtitles numerically, as Table 1 shows:

Data	Train	Dev	no len.	en len.
Gov.	50, 000	2,500	13.85	16.81
Sub.	250,000	2,500	6.16	6.76
Bok.	-	2,500	13.53	13.83
		Test		
DIY	-	5,000	6.70	6.90

Table 1: Dataset Statistics

3 Architecture

The Transformer Vaswani et al. (2017) consists of a stack of identical encoder and decoder layers. Each layer consists of a multi-head self-attention followed by feed-forward neural networks. The self-attention in the encoder is used to capture the dependencies between the words in the input sentence, while the self-attention in the decoder is used to capture the dependencies between the words in the output sentence generated and a cross attention which allows the decoder to attend to different parts of the encoder's output while generating the output sequence.

3.1 Baseline

Our baseline employs the typical transformer with default settings. Other configuratioins include:

Vocabulary size refers to the number of unique words or tokens present in a given text collection or dataset. We train the tokenizer with hard-coded vocabulary size. In our task, we have a subset of the original data, and we are trying to construct for two languages, English and Norwegian, we set the size to 10K rather than a common 30K.

Embedding Dimension defines the dimension of the input of the token embedding fed into the model. Common values range from 512 to 1024 dimensions, with larger sizes generally resulting in better performance at the expense of increased computation. Out of the same concern, we choose a lower embedding value of 256 for the dataset of this task.

For the sake of training time, we skip the step of searching for the optimal vocabulary size and embedding size, since our main focus is on the model size and inference time.

As requested, we train the model on three different datasets, and then we choose the best among all datasets as the baseline.

3.2 Parameter Sharing (PS)

Parameter sharing is a regularization technique used during the training phase to reduce the number of trainable parameters in a model, thereby speeding up training convergence and improving generalization. During inference, the parameters of the model are fixed and used to make predictions on new data, and parameter sharing by itself does not help reduce inference time at this stage.

However, parameter sharing can indirectly affect inference time by reducing the overall model size. Smaller models require less memory and can be loaded faster, which potentially speeds up inference. Furthermore, sharing parameters among model components can reduce memory footprint and computational overhead during inference, which may improve inference speed.

We specifically examine whether parameter sharing can effectively reduce the time required for inference, by checking the inference time of models with and without parameter sharing, we aim to determine if this technique provides any noticeable improvements in terms of computational efficiency during the inference phase.

In our default setting for the machine translation task, we've shared embeddings layer and output layer, which refer to using the same word embeddings for both source and target languages. By sharing embeddings, we control the total mount of parameters within the boundary of 10M, and the model can exploit the similarity between the

[1] http://data.europa.eu/88u/dataset/elrc_1061

[2] https://www.nb.no/sprakbanken/en/resource-catalogue/oai-clarino-uib-no-parallel-nob

[3] https://opus.nlpl.eu/OpenSubtitles-v2018.php

[4] https://farkastranslations.com/bilingual_books.php

[5] https://github.com/Helsinki-NLP/Tatoeba-Challenge

source and target languages, improve the efficiency of the model, and make it more robust to changes to variations in the training data.

Further, guided by previous studies on parameter sharing, we try to apply it to parts of the transformer. There are different strategies. In Sachan and Neubig (2018), the authors propose a method for learning a joint representation of sentences in multiple languages using shared word embeddings and a shared encoder-decoder model, and they show that their approach can improve translation performance for low-resource languages, and similar approach in Nayak and Ng (2020). Inspired by them, we try to share the self-attention layer between the two components, encoder and decoder, to explore the potential information transfer and its impact on enhancing the model's generalization capability.

Another strategy we try is that we keep the encoder unchanged and prioritize modifications in the decoder by introducing layer sharing and parameter sharing across all layers. We wonder if there is a significant difference between different layers within the decoder, and we expect an equal or loss of power or expressiveness since the decoder may or may not be able to learn the different features and patterns of the target language and struggle to generate fluent and accurate translations for complex sentences or difficult translations.

Here are the three variants of our NMT models.

- Baseline: No PS for either Encoder or Decoder.
- PS between Encoder and Decoder: We share the self-attention layer between Encoder and Decoder.
- PS within Decoder: We share the parameters among all layers within the Decoder.

3.3 Number of Encoder/Decoder Layers

The standard transformer architecture utilizes 6 encoder and decoder layers, but variations exist with up to 12 layers or fewer. More layers lead to a deeper understanding of context but increase computational requirements. Fewer layers most likely result in under-fitted models or slower convergence during training.

Kong et al. (2021) pointed out that for bilingual translation, using a deep encoder and shallow decoder (DESD) can reduce inference latency while maintaining translation quality. They further suggest using a multi-decoder architecture called Deep Encoder with Multiple Shallow Decoders (DEMSD), where each shallow decoder handles a non-overlapping part of the target languages. The DEMSD variant achieved a significant improvement in computation time, almost twice faster, without sacrificing translation quality, compared to traditional Transformer architectures.

Drawing inspiration from this, we adapt the approach to suit our specific scenario. We introduce an asymmetric architecture by adding more layers to the encoder and reducing an equal number of layers from the decoder. This design choice is motivated by the fact that constructing the decoder is computationally more expensive than the encoder, primarily because the decoder incorporates both self-attention and cross-attention mechanisms. By "strengthening" the encoder and "weakening" the decoder, we aim to achieve a more efficient model while reducing the overall size. This design helps balance computational resources and optimize the trade-off between model complexity and performance.

We explore three different setups as follows:

- Baseline: 6 layers for both Encoder and Decoder
- Balanced minus: 4 layers for both Encoder and Decoder
- Unbalanced: Deep (9, increase half) Encoder and Shallow (3, reduce half) Decoder

3.4 Pruning

Pruning is used to simplify large neural networks while preserving their performance, making them easier to interpret, store, and deploy. The goal of pruning is to remove redundant neurons or connections that don't contribute significantly to the model's predictions, thereby reducing computational complexity and energy consumption. See et al. (2016) shows the benefits of pruned models of an encoder-decoder sequential model, including reduced model sizes, accelerated inference speeds, and a lower risk of overfitting due to regularization effects.

Unlike dimensionality derivation in feature engineering, pruning targets weight sparsification to minimize the size of unimportant connections while retaining only essential information to maintain the desired level of predictive power.

3.4.1 Embedding and Final Classification

Word Embedding Compression aims to reduce the memory footprint of word embeddings without compromising their semantic information. It utilizes various methods such as quantization, clustering, or dimensionality reduction Kim et al. (2020) to achieve a more compact representation of word embeddings. However, pruning doesn't change the model structure so it's not meant to reduce the number of parameters, but to zero them out to save computation. We apply the pruning to the final classification layer (FC), as well as the embedding layer since they are sharing the parameters, we set it to 50% to see if weight elimination will result in better performance.

- Baseline: no pruning
- Pruning on Embedding and FC layer, up to 50%, we simply prunes random parameters.

3.4.2 Key, Query, Value

It's easy to think of applying pruning to the key, query, and value linear transformations as it can significantly reduce the corresponding computation of the self-attention and cross-attention and lead to lower memory requirements. This is particularly useful in scenarios where memory is a limiting factor, such as on mobile devices or in real-time applications. In later sections, we use pruned attention to refer to pruning keys, queries and values.

- Baseline: no pruning
- Pruning on Key, Query, Value: magnitude pruning, 50% components with the lowest L1 norm get masked

3.4.3 Attention Heads

Each layer within the encoder uses multi-head self-attention, and each layer within the decoder uses multi-head masked self-attention and cross-attention where each head refers to a specific way of computing attention weights. Typically, transformers have 8-12 heads per layer. The capacity of capturing global dependencies among tokens is affected when there are either too few or too many heads.

Voita et al. (2019) discovered that, that even after training models normally (with all heads), many heads can be removed at a test time, and it will not significantly affect the BLEU score, in fact, in some cases removing a few heads led to improving BLEU scores. We plan to employ a similar approach to estimate the importantness of all the heads during training, and eliminate non-essential heads in order to reduce processing time.

We devise an "attention" learning on the heads:

- Baseline: 8 heads
- Half size: it is not pruning but for comparison the effect of reducing the heads.
- Prune the heads: We aim to keep the important ones and zero out the least important ones for all layers.

We do not apply the complex way as Voita et al. (2019) suggested, instead, we approximate it by introducing additional parameters to assign weights to all heads, which only adds a very small amount of parameters, 144 to learn, then use the softmax weight to measure the weight of the heads. Ideally, we expect the weight of certain heads to be close to zero by the end of training. In this way, we can prune the least important attention heads to improve prediction speed.

4 Hypothesis

We posit that our baseline model, in its original structure, exhibits basic performance with the default parameter settings. Based on that, we explore multiple model compression techniques. We hypothesize that at least one of these configurations will surpass the baseline model, either by reducing the overall model size, or by improving the efficiency of inference, or achieving both of these objectives simultaneously.

5 Experiment and Result

We train for limited steps for all the settings, around 50K (10 epochs), and we use a greedy search to compute the BLEU score at the end of each epoch for all the validation datasets, including the government, the subtitles, and the book. Under our setting, it takes around 1-2 hrs for one complete round. We change one condition at a time e.g., layers, and freeze the rest of the settings, for a fair comparison. We report the overall performances in Table 2.

We observe a consistent trend of performance improvement as the number of steps increases for all the validation dataset. The highest performance is typically attained in the last epoch, although there are occasional instances where the peak performance occurs in the previous epoch. Notably,

during the last two epochs, we observe a flattening of the growth trend, indicating that further training beyond this point may yield diminishing returns in terms of performance improvement. In cases where the peak performance does not align across all three validation datasets, e.g. with the "book" dataset showing the best performance at the 9th epoch while the other two datasets reach their peak at the 10th epoch, we prioritize the "book" dataset as it represents an "out-of-domain" and real-world scenario, thereby demonstrating the model's generalizability. However, the overall impact is negligible since the differences in performance across the last two epochs are minimal.

It is also important to note that we present the figures starting from 30K steps til 50K steps to emphasize the differences between the various settings. However, when examining the results on a global scale, the discrepancies between the settings become less pronounced, with a maximum difference of around 2 BLEU scores. We set the acceptable tolerance of a decrease in BLEU score to be less than 1. Ultimately, it is crucial to consider the overall impact and practical implications when making decisions regarding model size and performance. The execution code can be found through Github [6].

5.1 Different Training dataset

We trained the baseline models with different training datasets:

- train on the government data
- train on the subtitles
- train on the combined dataset

Not surprisingly, the combined dataset achieve the best results, in terms of BLEU score among all the validation datasets. The model trained on the government dataset shows limited effectiveness on datasets other than its own, similarly, the model trained on subtitles is primarily suitable for the subtitles dataset itself. This suggests that the government dataset, while high-quality, lacks robustness when applied to real-world scenarios. On the other hand, subtitles may contain inaccuracies and noise since they are not thoroughly verified. Consequently, models trained on either dataset demonstrate less robustness when handling "out-of-domain" data.

[6] https://github.uio.no/jiebi/NMT

5.2 Parameter Sharing

Unexpectedly, we find that sharing all the layers within the decoder yields comparable performance to our baseline model, and in fact, it even slightly, by 1e-1, outperforms the baseline on the "book" dataset, as shown in Figure 1. This observation suggests that there may not be a significant difference within individual layers from the decoder. However, further investigation is needed to understand the underlying reasons behind it.

When sharing between the encoder and decoder, there is a decrease of approximately 1 BLEU score across the board. This indicates that the encoder and decoder have distinct roles and specialized functions that are not easily interchangeable. While information transfer between them is possible, it appears to have a light negative impact on overall performance.

It seems that having idenetial layers within the decoder emerges as the optimal choice because it shows robustness on the "book" and meets our tolerance threshold. However, sharing between encoder and decoder can also be considered as a viable alternative since it closely aligns with the baseline model.

5.3 Number of Layers

The unbalanced architecture of the transformer encoder and decoder model demonstrates its compatibility with our baseline, and it even exhibits a slight performance advantage over the baseline on the "book" dataset, as shown in Figure 2. This finding suggests that having a deeper encoder and a shallower decoder can potentially enhance certain aspects of the model's capabilities, such as capturing and encoding information more effectively.

Despite the potential benefits of reducing the number of layers, such as decreased model size and inference time, we observed a drop of 1.67 BLEU Score. This decrease exceeds our predefined tolerance threshold, making it unsuitable for our intended purposes. Although the reduction in model complexity and inference time is desirable, the resulting loss in translation quality outweighs these advantages. As a result, we have opted against this approach to ensure that our models maintain a satisfactory level of performance.

5.4 Number of Heads

The investigation into the number of heads has yielded intriguing results. It appears that the num-

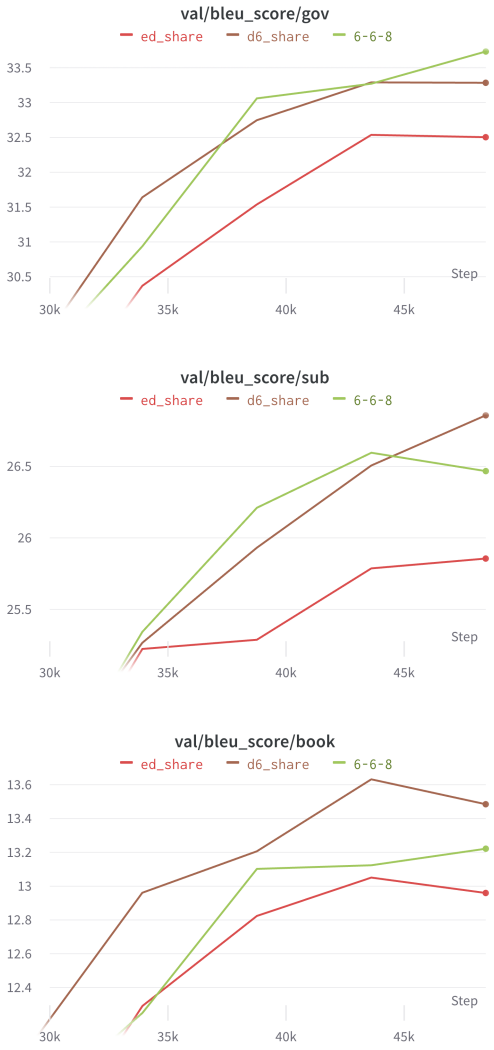


Figure 1: Parameter Sharing

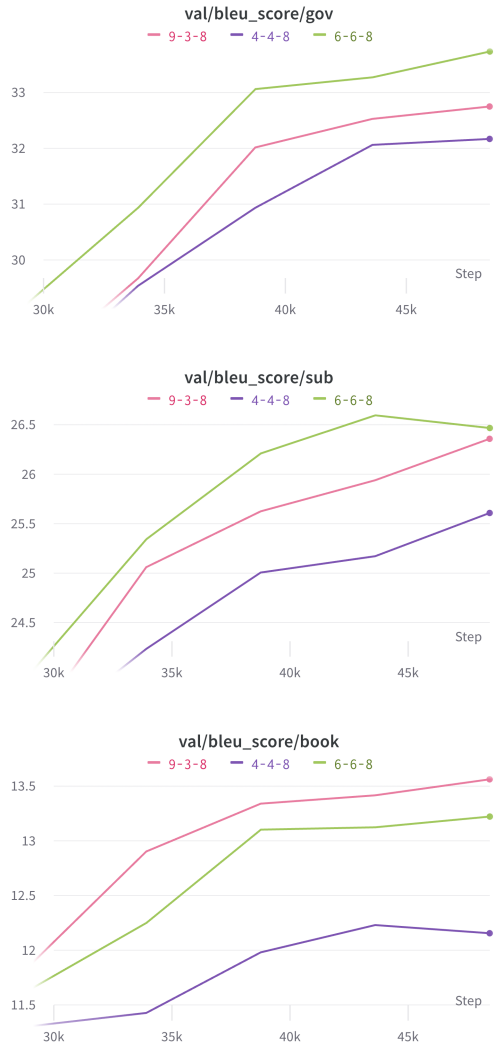


Figure 2: Number of Layers

ber of heads has no discernible impact on the performance of the model when evaluated on the government and subtitle datasets, as depicted in Figure 3, regardless of whether 8 or 4 heads are utilized. However, when pruning-head is applied, there is a light decrease in effectiveness across all cases. Furthermore, the performance of the model using only half the number of heads aligns closely with the performance observed when pruning heads on the "book" dataset.

Upon pruning heads, we observed a decrease of up to 2 BLEU score, more than our predefined tolerance threshold. This reduction in performance indicates that pruning heads negatively impacts the translation quality of our models. On the other hand, using half-heads yields results that closely aligned with the baseline performance on the government and subtitles datasets, with only a slight decrease in the book dataset, within our tolerance range. It also hints that the effectiveness of half-heads may vary depending on the specific dataset.

5.5 Pruning

Prunings fall slightly behind the baseline in terms of performance, albeit by a narrow margin. When considering the out-of-domain dataset "book", pruning embedding becomes nearly comparable to that of the baseline model. Overall, the variation in the BLEU score among schemes of pruning embedding is minimal, pruning attention is lightly over our threshold. Trimming the head, like discussed in Section 5.4 reduces the BLEU score of the government dataset by 2.

Here comes a question with the hidden size initialization. It may seem intuitive to start with a smaller embedding size to achieve model compres-

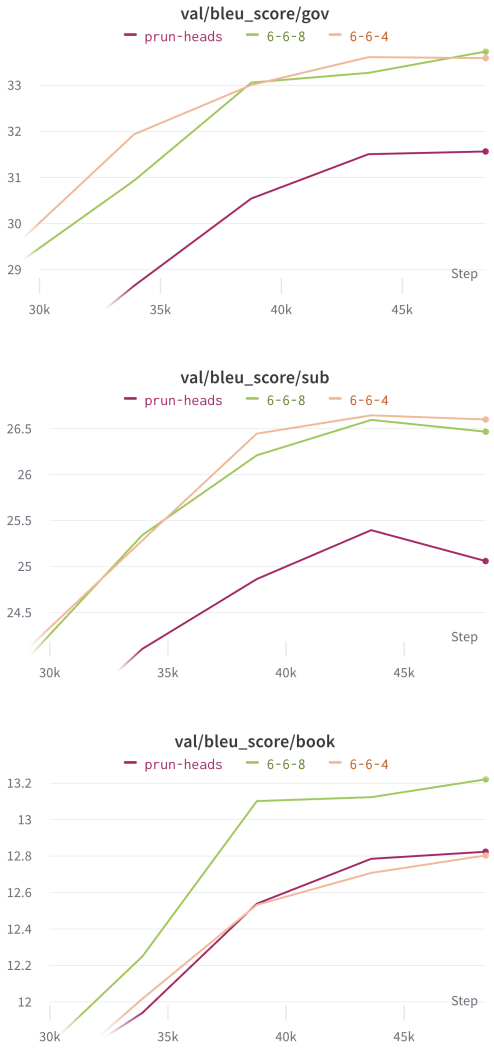


Figure 3: Number of Heads

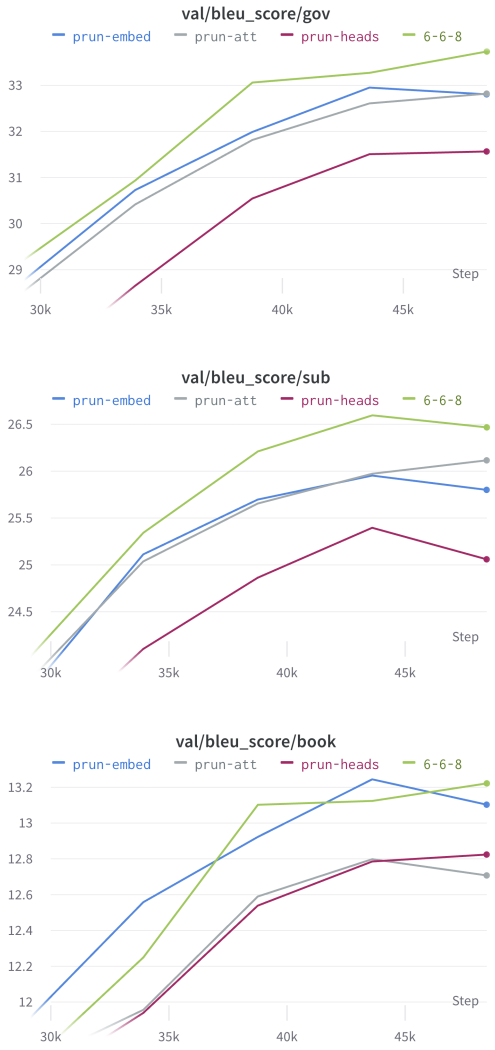


Figure 4: Different Pruning Schemes

sion, but it's also essential to consider the trade-off between model size and representation capacity. A smaller embedding size can indeed reduce the model's overall size, but it may also limit its ability to capture and encode the rich semantic and syntactic information present in the input data. Considering that we only cover a subset of the original dataset in this study, it is wise to keep the initial embedding dimension and prune it later.

5.6 Final Evaluation

We evaluate our models using our DIY dataset and measure their inference time in the CPUs on fox, simulating non-parallel resource devices. We perform this evaluation on all best checkpoints obtained from different settings. By recording the inference time, we evaluate the computational efficiency of the model and compare the performance of different configurations. This analysis provides insights into the real-world performance of our model and its suitability for deployment on devices with limited computational resources.

The obtained BLEU score on DIY is higher than the score on the validation datasets, which could be attributed to several factors. Firstly, it is possible that the content of the custom test dataset overlaps to a significant extent with the data used for training the model. This overlap in content allows the model to leverage its learned knowledge and patterns effectively, leading to better translation performance. Second, it could also be attributed to the utilization of beam search during decoding. Lastly, it can also suggest that the model exhibits a certain degree of generalization and is capable of generating translations that align well with the reference translations in the test dataset.

NO.	Model	Size	Train	Infer	BLEU				Acc.
	Combined		GPU	CPU	Gov.	Sub.	Bok.	DIY	
1	Baseline (668)	10,477,464	453.46	2506.99	33.73	26.47	13.22	38.29	-
2	SP in Dec.	6,523,830	451.97	2342.42	33.29	26.51	13.63	38.31	Y
3	Share between ED	8,904,600	446.23	2420.02	32.54	25.79	13.05	37.13	M
4	DESD (938)	9,686,400	407.79	1248.21	32.75	26.36	13.56	37.52	Y
5	prune embed	10,477,464	450.43	2369.10	32.95	25.95	13.24	37.55	Y
6	prune atten.	10,477,464	460.04	2222.64	32.61	25.97	12.80	37.78	M
7	prune heads	10,477,608	510.98	2481.42	31.57	25.06	12.82	36.63	N
8	Half Hds (664)	10,477,392	437.17	2101.66	33.59	26.60	12.80	38.88	Y
9	Half Lyrs (448)	7,841,984	326.43	1572.67	32.06	25.17	12.23	36.33	N
10	Half Lyrs & Hds (444)	7,841,936	316.36	1526.15	31.71	25.13	11.99	36.48	N
	Baseline with Sub.	10,477,464	303.97	-	6.05	25.77	10.92	-	-
	Baseline with Gov.	10,477,464	110.35	-	24.65	3.83	3.02	-	-

Table 2: The overall performance

We attach a column called "Acc." short for "Acceptance" to stands our model choices regarding their performance and efficiency upon all the datasets, mainly by the model size, performance in terms of BLEU score, and inference time for the Custom Dataset as reference only. In this context, "Y", short for YES, represents a preference choice, "M", short for "Medium" signifies a hesitant or uncertain choice, and "N", short for NO, indicates a denial or rejection choice, we rank the choices by our pre-defined threshold and all the validation and test datasets.

5.7 Inference Time

Theoretically, a model with a smaller size is expected to exhibit a lower inference time compared to our baseline model. However, it is important to note that reported inference times are indicative only and may vary due to various factors. These factors may include the presence of other concurrent tasks running on the CPU or the availability of system resources. Specifically, we have performed inference using all the models in parallel on a single computing node, where multiple tasks are competing each other for resources, consequently all jobs takes longer time to complete. Additionally, we have conducted sequential inference, one by one within one slurm file, where they are conducted within one computing nodes one after another, while it takes long to finish all tasks. We've also tried to submit jobs individually where tasks may be allocated to different computing nodes such as "C1-5" and "C1-17," alongside running tasks from others (their tasks have been allocated to the same computing nodes). We report the recorded inference time by the way of sequential execution in same computing nodes, and it meets our expectation: all the compressed model consume less time than the baseline. Nevertheless due to these variable conditions, it is challenging to provide an accurate and objective estimate of the speed or slowness under different model settings. The actual inference times may vary and are difficult to predict precisely.

6 Conclusion

In this study, we have undertaken several adaptations of the baseline model, focusing on the practical aspects. Among the various modifications, we have observed that several compressed models are favorable, including model 2,4,5,8. If we relax our threshold, model 3,6 are also viable solutions.

It is essential to acknowledge the limitations of our study. We have not undertaken a comprehensive and exhaustive exploration encompassing all potential configurations. Instead, we have deliberately chosen an extreme scenario, specifically pruning 50% of the parameters, to evaluate its impact on the model's output to a certain extent. Moreover, it is worth noting that experimental results can exhibit inherent randomness. To mitigate this, we have taken measures to control the randomness by setting a seed to ensure consistent results across experiments.

Considering the results obtained from pruning, there exists an opportunity to delve deeper into exploring various pruning schemes, e.g. pruning timing, throughout the whole training, in the early

stages, or while the training is close to converge Behnke and Heafield (2021). Additionally, exploring alternative attention mechanisms that are less computationally intensive, such as sparse attention Kitaev et al. (2020), linear attention Wang et al. (2020), or a combination of local and global attention Beltagy et al. (2020), holds promise. While these aspects were not specifically addressed in our study, since it is not worth exploring them in the context of short text data, where it typically imposes less computational burden. However, these directions offer significant potential for enhancing the practicality and efficiency of machine translation models when it handles long text, such an article, a whole book or an archive.

References

Maximiliana Behnke and Kenneth Heafield. 2021. Pruning neural machine translation for speed using group lasso. In *Proceedings of the sixth conference on machine translation*, pages 1074–1086.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

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.

Pieter Delobelle, Thomas Winters, and Bettina Berendt. 2020. RobBERT: a Dutch RoBERTa-based Language Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3255–3265, Online. Association for Computational Linguistics.

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

Yeachan Kim, Kang-Min Kim, and SangKeun Lee. 2020. Adaptive compression of word embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3950–3959, Online. Association for Computational Linguistics.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Xiang Kong, Adithya Renduchintala, James Cross, Yuqing Tang, Jiatao Gu, and Xian Li. 2021. Multilingual neural machine translation with deep encoder and multiple shallow decoders. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1613–1624, Online. Association for Computational Linguistics.

Anis Koubaa. 2023. Gpt-4 vs. gpt-3.5: A concise showdown.

Tapas Nayak and Hwee Tou Ng. 2020. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8528–8535.

Devendra Sachan and Graham Neubig. 2018. Parameter sharing methods for multilingual self-attentional translation models. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 261–271, Brussels, Belgium. Association for Computational Linguistics.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of neural machine translation models via pruning. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301, Berlin, Germany. 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, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Leveraging monolingual encoders and decoders to perform Neural Machine Translation

Lucas Georges Gabriel Charpentier
University of Oslo, Language Technology Group
lgcharpe@ifi.uio.no

Abstract

Given the difficulty of obtaining a large, high-quality parallel database, we want to create a model capable of achieving good translations with a limited amount of data of mixed quality. To this effect, we propose to exploit monolingual models' high performance to create a translation model. Our results show that we can achieve good performances, that still underachieve the state-of-the-art but require 30x less data and 60x less time to train. We also show that fine-tuning every part of the combined encoder-decoder model is not necessarily useful and that even training the bare minimum (only the cross-attentions) can lead to decent translation models. The code can be found on github[1].

1 Introduction

Great strides have been made in neural machine translation since the introduction of the Transformer (Vaswani et al., 2017). We now have automated translations that sound plausible to the point where we can use them to automatically translate websites. However, to achieve these results we generally need large amounts of parallel data which can be hard to come by.

In general, these large datasets have to be created manually and take a lot of human resources and time. In the cases where we try to automate the alignment of a source to a target language, we can end up with datasets of low quality. Which when used to train models, can lead to models that incorrectly translate or are incoherent.

On the other hand, getting monolingual data is much simpler. We can extract text from Wikipedia[2] using their archival database or scrap from websites (with permission from the website's author(s)). To then train models to either generate text or provide language representations (at inference time), we can use techniques such as Masked Language Modeling (Devlin et al., 2018) or causal language modelling. Since we can obtain more monolingual data, we can train better monolingual encoders, decoders, or encoder-decoder models.

In this paper, we propose combining a pre-trained monolingual encoder with a pre-trained monolingual decoder. We want to leverage the high performance in language representation and text generation of each model to create a good translation model when only a moderate amount of medium-quality data is available.

2 Background

Most neural machine translation models are based on the encoder-decoder structure where the encoder models the sense of the source text to translate while the decoder generates the translation based on the encoder's modelling. Since the Transformer (Vaswani et al., 2017) all modern and high-performing models use parallel attention rather than sequential attention which was the case in RNNs.

Even though using encoder-decoder models can model language, generate text or do translations, they are not necessarily the best when it comes to specific tasks. Also, by being trained for translation, it was not evident how to fine-tune them to perform another NLP task (such as NER). This is where the GPT models (Radford et al., 2018, 2019; Brown et al., 2020) came in. Instead of having both an encoder and decoder, one could just train a decoder using causal masking. This led to models that could be fine-tuned for a wide variety of NLP tasks and models that were good at text generation, whereas GPT models have shown to be proficient at generating coherent and correct text. This is further exemplified by the recent wide adoption of ChatGPT, a model based on the GPT architecture.

However, by going to a decoder-only model with causal masking, the model lost its bi-directionality

[1] link provided with final submission
[2] https://en.wikipedia.org

which made it good at solving tasks like sentiment analysis. Therefore, the authors in Devlin et al. (2018) decided to use the encoder from the Transformer rather than the decoder. This led to the creation of BERT, a bi-directional encoder-only model, which outperformed GPT in morpho-syntactic tasks such as NER or POS-tagging, sentiment analysis, and more where leveraging tokens both before and after the considered token could be useful.

3 Related Works

Other papers have thought of using pre-trained encoders or decoders to train a neural machine translation model. In Imamura and Sumita (2019), the authors use a pre-trained BERT as the encoder while training the decoder from scratch. They show that doing this does lead to better BLEU scores and potentially better translations. While we also use a pre-trained BERT as encoder, we also use a pre-trained decoder (GPT2) instead of training a decoder from scratch. This will reduce the total amount of training needed, since we do not need to start by pre-training the decoder.

In Weng et al. (2020), they test various different combinations of using a BERT and GPT model as either encoder or decoder. They also propose a new framework called APT which adds either a dynamic fusion mechanism or knowledge distillation or both to the pre-trained encoder and pre-trained decoder. They show that using a pre-trained encoder as an encoder and a pre-trained decoder as a decoder can lead to better results, especially when combined with the APT framework. While they do use pre-trained encoders and decoders, they pre-train them themselves using, for two of the three languages considered, data from the same source as the translation data. Instead of using a monolingual embedding, they use multilingual embeddings to pre-train the monolingual models. In addition, we do a more fine-grained fine-tuning analysis, considering training only certain parts of the whole model (for example only training the cross-attentions).

In Sun et al. (2021), they test using a multilingual BERT as an encoder and a multilingual GPT as a decoder and then add extra untrained encoder/decoder layers on each pre-trained model, called grafting layers. They show that fine-tuning the decoder is not necessary while fine-tuning the encoder is important. In addition, they show that their model (Graformer) outperforms a traditional Transformer in all tested languages. As for the previous paper, they pre-train there own multilingual BERT and GPT instead of using existing models. Contrary, to us who look at combining a pre-trained encoder and decoder with no extra layers, they add extra un-trained encoder and decoder layers which communicate with each other, while the BERT and GPT are used as extra encoding/decoding steps for the model.

4 Methods

Our proposed translation model is a combination of a pre-trained encoder and a pre-trained decoder. To achieve this we add cross-attention to each decoder layer. Then we pass the last hidden states of the encoder to each of these cross-attentions. In other words, if the original decoder layer (as in Radford et al. (2019)) can be defined as:

$$g(x) = s(l_1(x)) + x$$

$$\textit{DecoderLayer}(x) = f(l_2(g(x))) + g(x)$$

where l_1 and l_2 are LayerNorm layers, f is a feed-forward network, and s is self-attention. Then our new decoder layer is:

$$t(x) = c(l_3(g(x)), h) + g(x)$$

$$\textit{NewDecoderLayer}(x) = f(l_2(t(x))) + t(x)$$

where l_3 is a new LayerNorm layer, c is the cross-attention layer, and h is the last hidden states of the encoder.

In addition to creating this combined model, we also create variations by partially freezing some of the parameters of the model. This means that during fine-tuning the weights and biases of these parameters are not updated. We create four different variations of the model:

1. **Cross-attention only**: We freeze all pre-trained parameters from the encoder and decoder, and only fine-tune newly added cross-attentions.

2. **Cross-attention+decoder**: We freeze the encoder but leave the entire decoder unfrozen.

3. **Cross-attention+encoder**: We freeze the decoder apart for the newly added cross-attentions but leave the entire encoder unfrozen.

Hyperparameter	Value
Encoder layers	6
Decoder layers	6
Heads	8
Hidden size	256
Vocabulary size	8,000
Dropout rate	0.1
Number of Parameters	12M
Optimizer	AdamW
Max learning rate	0.001
Weight decay	0.1
Batch size	1024
Learning rate scheduler	inverse square root
Warmup steps	4,000
Number of steps	100,000

Table 1: Hyperparameters for the Baseline model

Hyperparameter	Value
Encoder	NorBERT3 (Base)
Decoder	GPT2 (Base)
Number of Parameters	276M
Optimizer	AdamW
Max learning rate	0.00008
Weight decay	0.1
Batch size	512
Learning rate scheduler	cosine
Warmup steps	2% max steps
Epochs	5
Seed	83947

Table 2: Hyperparameters for the NorBERT3+GPT2 model

4. **Full**: The whole model is unfrozen and fine-tuned.

Finally, we also train a small (less than 15M parameters) baseline model based on the Transformer architecture (with small improvements described in Section 5.2) to verify that our proposed model achieves good results.

5 Experiments

In this paper, we run two sets of experiments. The first experiment looks at whether using a pre-trained encoder and pre-trained decoder as described in Section 4 can results in a good neural machine translation model as compared to a baseline model. The second experiment looks at the effect of label smoothing (Szegedy et al., 2016) on our proposed model. To evaluate these models, we use the BLEU metric (Papineni et al., 2002). More specifically we will be using SacreBLEU (Post, 2018). This version wraps the original implementation of BLEU from Papineni et al. (2002) with extra features such that it is compatible with most tokenization and languages. Even though this metric is not effective at judging how well our models perform (Freitag et al., 2022), it is widely used and gives us some idea of how good models perform against each other. In addition, we do a quick qualitative look at the translations to have a better idea of how well each model performs.

During training, we evaluate our models using a dataset from a different source as a validation set. For the baseline model, we do this every 5 epochs, while for the NorBERT3+GPT2 model, we do this every epoch.

During validation, we generate translation by initializing our translation as the beginning of the sentence token and using greedy decoding to generate the full translation. In other words, at each step, we append the token, at the end of the predicted sequence, with the highest probability to our generated translation. We do this until the generated token is an end-of-sentence token or if our translation is 128 tokens long.

During testing, we generate translations by initializing our translation as the beginning of a sentence token and using beam search decoding to generate the full translation. As opposed to greedy decoding where we only keep to highest probability token at each step, with beam search we keep the top-k (size of the beam) translations until they all either end in an end-of-sentence token or are 128 tokens long. In our case, we use a beam of size 4.

5.1 Datasets

In this section, we will describe the datasets used for training, validating and testing the models.

Government Union of two openly available corpora: 1) Bilingual English-Norwegian parallel corpus from the Office of the Auditor General (Riksrevisjonen) wesite[3] and 2) Public Bokmål-English Parallel Corpus (PubBEPC)[4]. We then split this dataset with 50,000 training examples and 2,500

[3] http://data.europa.eu/88u/dataset/elrc_1061

[4] https://www.nb.no/sprakbanken/en/resource-catalogue/oai-clarino-uib-no-parallel-nob/

Model	BLEU			
	Government	Subtitles	Book	Tatoeba (2021-08-07)
$\text{Baseline}_{\text{government}}$	34.9	6.4	4.4	10.6
$\text{Baseline}_{\text{subtitles}}$	8.1	26.6	11.1	35.8
$\text{Baseline}_{\text{combined}}$	39.2	27.2	15.2	37.4
$\text{NorBERT3+GPT2}_{\text{cross-attention}}$	40.1	32.4	16.2	49.8
$\text{NorBERT3+GPT2}_{\text{cross-attention+decoder}}$	43.1	32.7	15.9	49.1
$\text{NorBERT3+GPT2}_{\text{cross-attention+encoder}}$	**50.1**	**36.0**	**20.3**	53.0
$\text{NorBERT3+GPT2}_{\text{full}}$	**50.1**	35.2	**20.3**	51.2
OPUS-MT-no-en/2020-05-22*	-	-	-	**61.4**

Table 3: BLEU score of various models on 4 different datasets. For the Baseline models, the subscript indicates the training dataset used to train the model. The combined dataset is the union of the subtitles and government datasets. The NorBERT3+GPT2 models were trained on the combined dataset and the subscript indicates which parts of the model were fine-tuned. *results from https://opus.nlpl.eu/leaderboard/ for nob-eng translations.

validation examples. While this dataset is not very big, it is of high quality since the text come from official government translations.

Subtitles An aligned Bokmål-English dataset done by Lison and Tiedemann (2016) released to the public[5] based on the unofficial subtitles from movies and TV series published to OpenSutitles[6]. We then split this dataset into 250,000 training examples and 2,500 validation examples. This dataset is of medium size but the quality is only moderate. The translations are unofficial and in subtitles, the translation can be more figurative than literal.

Combined This is the combination of the government and subtitles training sets (300,000 parallel sentences).

Book This dataset contains the translated and aligned book "The Hound of the Baskervilles" by Sir Arthur Conan Doyle. This is part of the "Bilingual books" collection by FarkasTranslation.com[7]. This dataset is used exclusively for validation and contains 2,500 parallel sentences. We use it for validation because it is of good quality and in a very different domain from the training datasets. Given this, we can test how our model generalizes.

Tatoeba Dataset of parallel Bokmål-English sentences derived from Tatoeba[8] compile by Tiedemann (2020). We use the test set from the 7th of August 2021, which contains 4,539 parallel sentences. This dataset contains more general translations as well as some expression translations. Some of the examples can also be multi-sentence.

[5] https://opus.nlpl.eu/OpenSubtitles-v2018.php

[6] http://www.opensubtitles.org/

[7] https://farkastranslations.com/bilingual_books.php

[8] https://tatoeba.org/

5.2 Models

In this paper we use two different models, the first is a baseline model based on the Transformer (Vaswani et al., 2017) architecture with a few improvements. In our implementation, we use "pre-norm" from Nguyen and Salazar (2019) instead of the traditional "post-norm", the "position-infused attention" idea from Press et al. (2021) in addition to the absolute positional encoding from the Transformer paper, and the GLU non-linearity from Shazeer (2020) instead of ReLU. These improvements both improve the performance and stability of the Transformer model. The hyperparameters for these baseline models can be found in Table 1, it also includes the training hyperparameters.

Our second model is a union of a pre-trained NorBERT3 from Samuel et al. (2023) as the encoder and GPT2 from Radford et al. (2019) as the decoder. To combine the encoder and decoder into an encoder-decoder model, we add cross-attention sub-layers to each layer of the pre-trained GPT2 model. This can easily be achieved with the HuggingFace transformers library (Wolf et al., 2020). Using the GPT2 pre-trained model from their models' library, we can change its configuration so that includes untrained cross-attention layers. We then set the weights of all the other layers to be equal to those of the pre-trained model. We pass the last hidden states from NorBERT3 to the cross-

Model	Output
Source Text	Er det ett ting historien har lært oss, så er det at vi ofte spår feil om fremtiden.
$\text{Baseline}_{\text{combined}}$	If there's one thing the story has taught us, it's that we often be eating the wrong about the future.
$\text{NorBERT3+GPT2}_{\text{cross-attention}}$	If there is one thing that history has taught us, it is that we often underestimate the future.
$\text{NorBERT3+GPT2}_{\text{cross-attention+encoder}}$	If there's one thing the history has taught us, it's that we often make wrong predictions about the future.
Target	If there is one thing history has taught us, it's that we often incorrectly predict the future.

Table 4: An example translation (from the tatoeba set) for 3 different models, the best baseline model, NorBERT3+GPT2 trained on cross-attention and NorBERT3+GPT2 trained on cross-attention+encoder.

attention of each decoder layer as would be the case in a standard encoder-decoder. The general model and training hyperparameters for this model can be found in Table 2.

Based on those two general models, we train three different baseline models, dependent on the training dataset used for training, and four NorBERT3+GPT2 models based on which parts we fine-tuned as described in Section 4. All the NorBERT3+GPT2 models are trained on the dataset which results in the best baseline model.

$\text{Baseline}_{\text{government}}$ A Baseline model trained on the government training set.

$\text{Baseline}_{\text{subtitles}}$ A Baseline model trained on the subtitles training set.

$\text{Baseline}_{\text{combined}}$ A Baseline model trained on the combined training set.

$\text{NorBERT3+GPT2}_{\text{cross-attention}}$ A NorBERT3+GPT2 model where only the cross-attentions are fine-tuned.

$\text{NorBERT3+GPT2}_{\text{cross-attention+decoder}}$ A NorBERT3+GPT2 model where the cross-attentions and the decoder (GPT2) are fine-tuned.

$\text{NorBERT3+GPT2}_{\text{cross-attention+encoder}}$ A NorBERT3+GPT2 model where the cross-attentions and the encoder (NorBERT3) are fine-tuned.

$\text{NorBERT3+GPT2}_{\text{full}}$ A NorBERT3+GPT2 model where all parameters are fine-tuned.

Our Baseline models took between 1h45 and 2h to train on a single A100 Nvidia GPU. The NorBERT3+GPT2 models took from 1h20 (for $\text{NorBERT3+GPT2}_{\text{cross-attention}}$) to 2h (for $\text{NorBERT3+GPT2}_{\text{full}}$) to train on a single RTX3090 Nvidia GPU.

5.3 Results

We begin by training the baseline models on the three different datasets. The dataset that leads to the best baseline model is then used to train each variation of the NorBERT3+GPT2. The results for all these models can be found in Table 3.

5.3.1 NorBERT3+GPT2 versus Baseline

If we start by looking at the results for the baseline models concerning the dataset used to train, we see that using the combined dataset results in the best model. This is the case on all validation and test sets. If we focus on the other two models, we see that they achieve very similar scores to the combined model when considering the validation set taken from the same source as their training set, but do significantly worse in the case of $\text{Baseline}_{\text{government}}$ or moderately worse in the case of $\text{Baseline}_{\text{subtitles}}$ (except for the government dataset where it performs significantly worse) on the other datasets. Based on these results we train our NorBERT3+GPT2 models on the combined dataset.

Comparing $\text{Baseline}_{\text{combined}}$ and $\text{NorBERT3+GPT2}_{\text{cross-attention}}$ we see that $\text{NorBERT3+GPT2}_{\text{cross-attention}}$ outperforms the baseline on every set. In addition, it takes 30-40 minutes less time to fine-tune this model as compared to training the baseline from scratch.

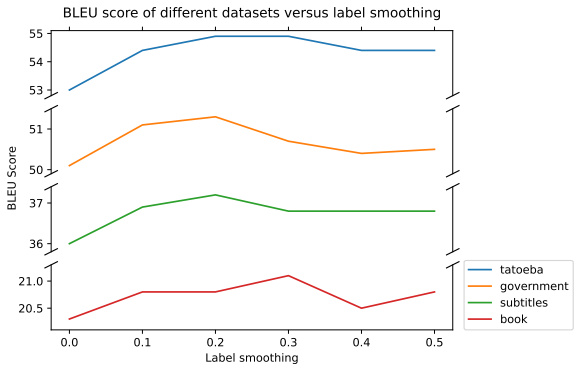


Figure 1: BLEU score for each set dependent on label smoothing. The model used is a NorBERT3+GPT2$_{\text{cross-attention+encoder}}$.

While the gains on the government, subtitles and book datasets are modest, the BLEU score on the tatoeba dataset increases by 12.8 points or a 34% increase in score. This is in line with our hypothesis that fine-tuning the cross-attentions between a pre-trained encoder and a pre-trained decoder can lead to a decent translation model. However, one thing to note, is that the NorBERT3+GPT2 models are about 20x larger than the baseline models. Here, we wanted to compare models that would take roughly the same amount of time/FLOPS to train rather than models of equivalent sizes. It is possible that if we made a baseline model with the same number of parameters as our NorBERT3+GPT2 model, the former would outperform the latter.

If we then focus on the different variants of the NorBERT3+GPT2 model, we observe two different phenomena. The first is that fine-tuning the decoder does not improve the model (as in Sun et al. (2021)) and in most cases makes it worse. NorBERT3+GPT2$_{\text{cross-attention+decoder}}$ has 0.3 less BLEU score on the book dataset and 0.7 less on the tatoeba dataset as compared to NorBERT3+GPT2$_{\text{cross-attention}}$. NorBERT3+GPT2$_{\text{full}}$ achieves similar performances as NorBERT3+GPT2$_{\text{cross-attention+encoder}}$ on the book dataset, while having a 1.8 lower BLEU score on the tatoeba dataset. For the government and subtitles datasets, we see the opposite trend, however, this is potentially due to the model overfitting on the style of the training data.

The second observed phenomenon is that fine-tuning the encoder is very important to improve the model. NorBERT3+GPT2$_{\text{cross-attention+encoder}}$ performs significantly better on all validation and test datasets compared to NorBERT3+GPT2$_{\text{cross-attention}}$. We hypothesize that, in this setting, the decoder is already good at generating text and does not need further refinement, but rather needs the signals given by the cross-attention to be of good quality so that it can map the source language to the target language.

Finally, we also add the best performing model (OPUS-MT-no-en/2020-05-22) on the tatoeba dataset to see how our model does compare to the state-of-the-art. We see that our model underperforms by 8.4 points, even though it is larger. However, the OPUS model is trained on 11,179,800 parallel sentences and took over 5 days to train on four GPUs. That is 30x more data and 60x more time (or 240x times more given the four GPUs versus our one GPU) to train. It is possible that if we used a slightly larger dataset, we could match or outperform the OPUS model while taking less time to train. In addition, parts of the tatoeba dataset have more than one sentences or are longer than 128 tokens. Since the training data for our model are relatively short sentences and only contain one sentence per example, our model might struggle to generate longer, multi-sentence translations.

Table 4 shows one example translation for the same source text of the Baseline$_{\text{combined}}$, NorBERT3+GPT2$_{\text{cross-attention}}$, and NorBERT3+GPT2$_{\text{cross-attention+encoder}}$ models.

If we start by looking at the Baseline$_{\text{combined}}$ translation, we see that it does some grammatical mistakes "...one thing the story..." or "... we often be eating...", confuses "history" with "story" and incorrectly translates the second part of the sentence "it's that we often be eating the wrong about the future." instead of "it's that we often incorrectly predict the future.". Overall, the translation makes little sense and while we would be able to deduce the true translation, it would require some creative thinking on our part.

NorBERT3+GPT2$_{\text{cross-attention}}$ and NorBERT3+GPT2$_{\text{cross-attention+encoder}}$ do a much better job of translating the sentence. While NorBERT3+GPT2$_{\text{cross-attention+encoder}}$ translation is close to perfect (minor grammatical mistake "...one thing *the* history..."), in terms of sense (the wording is not the same, but the same message is conveyed).

Label Smoothing	Output
Source Text	Etteraping av gester, handlinger og væremåter kan, men behøver ikke være, en bevisst handling.
No label smoothing	The reemergence of gestures, actions and ways of behaving may, but does not necessarily, be an intentional act.
0.1	The appreciation of gestures, actions and ways of behaving can, but does not have to be, a conscious act.
0.3	The re-emergence of gestures, actions and ways of behaving can, but does not need to be, a conscious act.
Target	Imitation of gestures, actions and manners may or may not be a conscious act.

Table 5: An example translation (from the tatoeba set) for different values of label smoothing, the model is a NorBERT3+GPT2$_{\text{cross-attention+encoder}}$.

NorBERT3+GPT2$_{\text{cross-attention}}$ confuses "incorrectly predict" with "underestimate" which while not exactly wrong (in certain contexts they can be synonyms), here the meaning of the second part is erroneous even though it is not very hard to deduce.

5.3.2 Effects of label smoothing

To test the effects of label smoothing on our NorBERT3+GPT2 model, we test 6 different values of label smoothing; 0, 0.1, 0.2, 0.3, 0.4, and 0.5. We take our best-performing model from the previous experiment, NorBERT3+GPT2$_{\text{cross-attention+encoder}}$ and train it to fine-tune it with each value of label smoothing using the combined training dataset and the hyper-parameters found in Table 2. The results of this experiment can be seen in Figure 1.

We see that having label smoothing has a moderate but noticeable effect on the BLEU score. As we increase the label smoothing, we increase our BLEU score on the tatoeba and book datasets until reaching a maximum at 0.3 label smoothing (54.9 for the former and 21.1 for the latter). After that the BLEU score decreases, this is potentially due to the model not being able to capture translation as well or not being trained for enough epochs. For the government and subtitles datasets, the model trained with 0.2 label smoothing, but since the training data are in the style as these datasets this is less informative on the translation ability of the model as compared to the tatoeba and book datasets.

While the maximum is reached with a label smoothing of 0.3, the largest improvement happens when going from no label smoothing to 0.1 label smoothing. For the tatoeba dataset, we have an absolute increase of 1.4 in the BLEU score and for the book dataset, the absolute increase is 0.5. This is in line with what we would expect since the goal of label smoothing is the reduce the over-confidence of networks.

These improvements can also be seen when looking at actual translations, Table 5 shows one such example.

We see that all three translations are not perfect and very similar. However, when there is no label smoothing the model uses "intentional" instead of "conscious" and uses "...may, but does not necessarily, be..." rather than "...can, but does not have/need to be,...". Both, especially using "intentional", give a more formal/professional tone. Using 0.1 and 0.3 leads to translations with only one difference "re-emergence" instead of "appreciation". In all three cases, the sense of the translation is quite correct, even if the wording is different.

6 Conclusion

As seen in Section 5 combining a pre-trained encoder and pre-trained decoder into an encoder-decoder model can lead to a good neural machine translation model. This is the case even when we

only fine-tune the added cross-attention while keeping the rest of the model parameters frozen. This hints that the model can leverage the language representations of each model to generate good translations.

In addition, we saw that fine-tuning the decoder would lead to either the same results or worst results than not fine-tuning it. While, for the encoder, fine-tuning it leads to better results. We hypothesize this is due to the decoder already being good at generating text in the target language and only needing to integrate the cross-attentions to be able to know what to generate rather than needing it to model language.

Finally, even though we were using a relatively small dataset with either a very specific style of language (in the case of the government dataset) or questionable translations (in the case of the subtitles dataset). We saw that we were able to create a model that does reasonably well on a more general dataset as compared to a model trained on a much larger and varied dataset.

7 Future Works

In future works, we would like to explore four different areas of interest:

- Using a higher quality and varied, but still small, dataset to perform the fine-tuning. With this, we could explore the potential of requiring smaller amounts of higher-quality data to be able to generate a good translation model.
- Comparing our model to a pre-trained model with the same amount of parameters.
- Instead of using a pre-trained encoder and pre-trained decoder, we would use the encoder part of one encoder-decoder model and the decoder part of another encoder-decoder model. Since the cross-attentions are already included in the decoder, we could have a better translation model.
- Norwegian has a lot of variation due to the wide range of dialects found in the country. Therefore using Nynorsk data as well as Bokmål could prove useful for better translations.

References

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.

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*.

Markus Freitag, Ricardo Rei, Nitika Mathur, Chi-kiu Lo, Craig Stewart, Eleftherios Avramidis, Tom Kocmi, George Foster, Alon Lavie, and André F. T. Martins. 2022. Results of WMT22 metrics shared task: Stop using BLEU – neural metrics are better and more robust. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 46–68, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Kenji Imamura and Eiichiro Sumita. 2019. Recycling a pre-trained bert encoder for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 23–31.

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).

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

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

Ofir Press, Noah A. Smith, and Mike Lewis. 2021. Shortformer: Better language modeling using shorter inputs.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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.

Noam Shazeer. 2020. Glu variants improve transformer.

Zewei Sun, Mingxuan Wang, and Lei Li. 2021. Multilingual translation via grafting pre-trained language models. *arXiv preprint arXiv:2109.05256*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Rongxiang Weng, Heng Yu, Shujian Huang, Shanbo Cheng, and Weihua Luo. 2020. Acquiring knowledge from pre-trained model to neural machine translation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9266–9273.

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