Comparative Analysis of Named Entity Recognition in Nordic Languages

Hadi Salameh
IT-University Of Copenhagen / NLP Exam 2021
hads@itu.dk

Abstract

Recent advances with NLP have made it possible for data engineers to develop Named Entity Recognition (NER) models to help us identify keywords in a sentence. In this paper, we will investigate how well a Danish NER corpus performs on different Bidirectional Encoder Representations from Transforms (BERT) Models. The BERT models used in the training are all based on Nordic languages. The paper found that the Danish corpus performed better on the Norwegian model due to it having been trained on a much larger corpus, than the Danish model, while the other BERT models did not outperform.

1 Introduction & Background

Words can have different meanings depending on the sentence context. For example, should Rundetårn (lit. trans. Roundtower) be understood as a general round tower or a location in Copenhagen? The task of Named Entity Recognition (NER), can help us identify entities in a sentence from a selected list of tags.

Transformers, such as BERT, have had very good success in NER-tagging. BERT(Devlin et al., 2019) is one of the most used transformer models in the NLP community due to it being free to use by individuals and companies.

This has given a chance for data engineers who don't have access to an enormous amount of computational power, to develop their own solutions with pre-trained models. More recently, the Danish NLP community has grown larger with multiple tagged corpus and pre-trained models popping up (Enevoldsen et al., 2021). More recently the Danish Alexandra Institute developed DaNLP, a repository with a collection of multiple datasets and models in Danish for multiple NLP tasks, with one of them being NER-tagging (Brogaard Pauli et al., 2021).

However, the Danish language shares many linguistic characteristics with other Nordic languages such as Norwegian and Swedish due to geographical position and historical reasons. In this paper, I will investigate if using pre-trained BERT models with nordic-languages other than Danish such as Norwegian, Icelandic, Swedish and Finnish performs close in NER-tagging compared to just using a pre-trained model on Danish.

Using the results from the comparison, I can determine if using a pre-trained model for another language with similar linguistic characteristics can be efficient. This can be useful for languages, in which the corpuses associated with it, don't contain enough data to train a model from scratch.

2 Similar work

Similar work has been done in regards to multilingual training. The paper (Hvingelby et al., 2020) describes their dataset DaNE. A corpus specifically set for NER and POS tagging in Danish. This is also the dataset I will be using for the NER task. In the paper, the authors compared different network architectures, datasets and language combinations to see which one performed better. The results showed that although the other models scored very high marks, a combination of a pre-trained Danish BERT model with a Danish training set outperformed all the others in most cases.

3 Methods

3.1 Tools

Although I am using pre-trained BERT models, I will still be fine-tuning it on the Danish NER corpus. Luckily, data engineers at *Ekstra Bladed* have been developing a tool called NERDA (NER

for Danish) (Kjeldgaard and Nielsen, 2020), which provides a user-friendly interface to fine-tune transformer models such as BERT, to a specified corpus.

3.2 Pre-trained BERT models

All the BERT models were downloaded from HuggingFace. A list of all the BERT models used in the training can be seen in table 1.

BERT-Model	Description
Finnish	Finish BERT trained by the University of Turku [1]
Icelandic-NER	Icelandic BERT trained by the Reykjavik University[2]
Multi-Languages	A multilingual BERT model trained by the HuggingFace team.[3]
Swedish	Swedish BERT trained by The National Library of Sweden[4]
Swedish-NER	Swedish BERT trained by The National Library of Sweden. It is specifically trained for the NER-task[5]
Dane-NER	Danish BERT created by BotXO.ai. It is specifically trained for the NER-task [6]
Norwegian	Norweigan BERT trained byt The National Library of Sweden.[7]

Table 1: BERT Models

3.3 Baseline

As a baseline, I will be using the result from the Danish pre-trained model, since that's what I am trying to get close to. We want to see if other Nordic pre-trained models perform close or even better than just using a Danish pre-trained model, so this will serve as a good baseline.

[1]https://huggingface.co/TurkuNLP
[2]https://huggingface.co/m3hrdadfi/icelandic-ner-bert
[3]https://huggingface.co/bert-base-multilingual-uncased
[4]https://huggingface.co/KB/bert-base-swedish-cased
[5]https://huggingface.co/KB/bert-base-swedish-cased-ner
[6]https://huggingface.co/Maltehb/danish-bert-botxo-ner-dane
[7]https://huggingface.co/NbAiLab/nb-bert-large

3.4 Hardware

The system used for the training contains a 3090 RTX, 64GB RAM and a 5950X CPU.

3.5 Dataset

The corpus I am working with is maintained by DaNLP, and called *Dane*[8]. However, I am using the NERDA API to download the corpus. It contains annotated data for danish sentences. The annotation includes POS and NER tags, but I will only be working with the NER tags. The tags follow the same structure as CoNLL (Another NER corpus for English). The possible values are shown in table 2

Tag	Description
O	Unknown/Padding
B-PER	Person Name (Start of Phrase)
I-PER	Person Name (non-initial word)
B-ORG	Organization (Start of Phrase)
I-ORG	Organization (non-initial word)
B-LOC	Location (Start of Phrase)
I-LOC	Location (Start of Phrase)
B-MISC	Miscellaneous Names (Start of Phrase)
I-MISC	Miscellaneous Names (non-initial word)

Table 2: Table listing tags

A simplified example of a DaNE dataset row can be seen in table 3

Text	NER-Tags	NER-Tags Ids
Hans bor i København	[B-PER, O, O, O, O, O, I-LOC]	[1, 0, 0, 0, 0, 0, 6]

Table 3: Partial example of a dataset row

3.6 Data Pre-processing

Before the training, all text and NER-tags needs to be processed. First, the sentence needs to be tokenized. For this, NERDA uses the HuggingFace Transformer API to convert it to tokens. The model used for tokenization is the same as the BERT model used in training. Secondly, we define a maximum sentence length. For simplicity, I have chosen 128 as the length. I use this number to either pad or crop the list of tokens, depending on whether it exceeds the maximum length. The BERT start and end tokens are also added to the list of tokens. Then at last the tokens are encoded into unique ids. The NER-Tags already comes as a list of unique ids, so they require little work. It should

[8]https://github.com/alexandrainst/danlp/blob/master/docs/docs/datasets.md#dane

be said, that all the text-processing is handled by NERDA.

3.7 Training

The dataset was already split into training and testing subsets. Unfortunately, DaNE does not include a validation subset, so I decided to take the training subset and then split it into 80% training and 20 % validation. The training set ended being of size 3506, the validation 877 and the test size 565. The training subset is mini-batched into size 32, while the validation and test subsets were batched into 16 and 8 respectively. NERDA does not currently support Early Stopping, so I opted for 15 epochs instead. A full overview of the hyper-parameters and settings can be seen in table 4

Setting	Value
Train Batch Size	32
Test Batch Size	16
Val Batch Size	8
Learning Rate	0.0001
Optimizer	Adam
Loss Function	Cross Entropy

Table 4: Settings

3.8 Evaluation

Using accuracy as a metric will not be sufficient for the evaluation. Accuracy is often used in cases where we have a balance of classes in our dataset. However, when working with a corpus, and especially NER, we cannot guarantee an equal or similar distribution of NER tags. Two other metrics called Precision and Recall can be used instead.

3.8.1 Precision

Precision tells us how many true positive we have out of the total number of entities that were predicted to be positive:

$$Precision = \frac{(TP)}{(TP + FP)}$$

It is useful when we want to penalize wrongly classified NER-tags (Huilgol, 2019).

3.8.2 Recall

Recall tells us how many true positives we have out of the actual number of positive cases:

$$Recall = \frac{(TP)}{(TP + FN)}$$

Recall can be used to as a measure of how many NER-tags where not being identified.

3.8.3 F1-score

Often we want to have a balance between Precision and Recall, and we therefor use F1-score as a measurement, which is the *Harmonic Mean* between Precision and Recall. A perfect balance is when Precision and Recall is equal. In the result section, I will be using the F1-score as a measurement of how well the models perform (Huilgol, 2019).

4 Results

The results from the test evaluation can be seen in table 5. It contains all the calculated F1-scores for each tag per, for each pre-trained model. As seen in the table, the Norwegian BERT model performs better for most of the tags than all the other language models, with the Danish model (baseline) coming in second place. All models seemed to perform very bad on I-LOC, due to many false negatives.

BERT-Model	B-PER	I-PER	B-ORG	I-ORG	B-LOC	I-LOC	B-MISC	I-MISC
Dane-NER	**0.95**	0.97	0.71	0.65	0.84	0.33	0.68	**0.82**
Finnish	0.82	0.86	0.54	0.44	0.63	0.19	0.55	0.55
Icelandic-NER	0.9	0.94	0.66	0.69	0.8	0.18	0.72	0.72
Norweigan	**0.95**	**0.99**	**0.78**	**0.78**	**0.87**	**0.38**	**0.79**	0.78
Multi-Languages	0.94	0.97	0.69	0.65	0.84	0.32	0.72	0.63
Swedish	0.89	0.93	0.69	0.6	0.75	0.25	0.68	0.61
Swedish-NER	0.88	0.94	0.61	0.52	0.75	0.32	0.68	0.38

Table 5: F1-Score tabel. Purple is the Baseline model

5 Discussion

The language that shares the most linguistic properties with Danish is Norwegian and it should therefore not be a surprise that the results from both are very close. What is surprising is that the Norwegian model outperformed the Danish NER-model, even though it was not trained for NER-tasks. The Norwegian model is created and maintained by the National Library of Norway, and therefore have access to access to a lot of documents. They have trained on documents from the last 200 years, which sums up to 109 GB [9]. The Danish NER model was trained by BotXO.ai[10]

[9] https://github.com/NBAiLab/notram

[10] https://github.com/certainlyio/nordic_bert

and is trained on a corpus size of 9.5 GB of text. This does give a huge advantage to the Norwegian model, and that must have played a role in the results. Nevertheless, this does show that in cases where we don't have enough data to train on, we can use other models from a similar language like the Norwegian model, which is what I wanted to investigate.

As mentioned in the result section, all models performed badly in the I-LOC tag. While I can't say for sure why it did not perform well on this tag, it probably has something to do with its representation in the dataset. To help the investigation, I created a distribution over the NER-tags as seen in fig. 1. The I-LOC tag is not represented well in the corpus. Oversampling could have been a temporary solution, to increase the number of examples. However, while that would make the model less biased towards the other tags, it still would not give a good representation of all the different ways I-LOC can appear in a sentence. Therefore the corpus needs to be updated with more text.

The worst performing model was the one based on the Finnish language. While it is considered Nordic, it does not share many linguistic properties with Danish, since it is a Uralic language and not a Germanic one like Danish. This explains why it did not score very high marks.

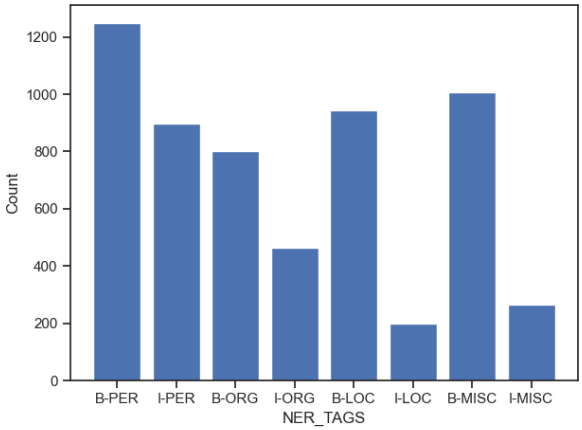

Figure 1: Caption

6 Conclusion

Based on the results, I can conclude that using a BERT language from a similar language to the language of the corpus, works well for the NER-task. It can help in cases, where the corpus size is not sufficient. However, for this to be true, the two languages must be very closely related to each other.

It is also very important for the distribution of classes to be balanced, and that the corpus used in the project needed to grow larger and with more examples of the I-LOC tag.

References

Amalie Brogaard Pauli, Maria Barrett, Ophélie Lacroix, and Rasmus Hvingelby. 2021. DaNLP: An open-source toolkit for danish natural language processing. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa 2021)*.

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

Kenneth Enevoldsen, Lasse Hansen, and Kristoffer Nielbo. 2021. DaCy: A Unified Framework for Danish NLP.

Purva Huilgol. 2019. Accuracy vs. f1-score.

Rasmus Hvingelby, Amalie Brogaard Pauli, Maria Barrett, Christina Rosted, Lasse Malm Lidegaard, and Anders Søgaard. 2020. DaNE: A named entity resource for danish. *LREC 2020 - 12th International Conference on Language Resources and Evaluation, Conference Proceedings*, (May 2020):4597–4604.

Lars Kjeldgaard and Lukas Nielsen. 2020. Nerda. GitHub.