Ælæctra

A Step Towards More Efficient Danish Natural Language Processing

Malte Højmark-Bertelsen

Study no.: 201809226

Supervised by Riccardo Fusaroli

Bachelor Thesis, Cognitive Science, Aarhus University

2020-01-06

Table of Contents

1 Abstract

The development of large Transformer-based language models (LMs) has caused an incremental need for more resources, including computational power and data, and has been reported to consume huge amounts of energy, giving rise to environmental concerns. These LMs are mostly developed for the English language, which leaves the technological capabilities of low-resource languages, such as Danish, in disadvantageous positions. Recently, however, researchers have found more efficient ways of training LMs and new Danish Natural Language Processing (NLP) resources are being collected. This paper introduces *Ælæctra* - an ELECTRA-Small pretrained on the Danish Gigaword corpus. The language capabilities of Ælæctra are evaluated by using two different Danish Named Entity Recognition (NER) datasets. Ælæctra performs similarly or slightly worse than previous state-of-the-art (SOTA) models, while being more than 3 times faster per epoch and using 8 times fewer trainable parameters. Thus, Ælæctra allows for rapid experimentation and is favorable to employ when increased model efficiency is of priority. Due to varying model performances on the two datasets, and because of the experimental setup only including NER as an evaluation task, this paper only demonstrates a limited picture of the general language capabilities of Ælæctra. Further testing should, thus, be prioritized to present a representative analysis of Ælæctra's language capabilities. To further enhance Danish NLP, a future focus should be on improving the collection of Danish NLP resources, including high-quality datasets and evaluation tools. Additionally, pretraining an Ælæctra with the ELECTRA-Base structure could benefit the current capabilities of Danish NLP. This paper further suggests to research the issues regarding replication of performance, and that the NLP community, in general, should continue to develop more ways of training efficient LMs.

Keywords: Danish, Natural Language Processing, Named Entity Recognition, Ælæctra, ELECTRA, BERT

2 Introduction

Natural Language Processing (NLP) is a field of study within the research of artificial intelligence (AI) and is concerned with how to enable computers to process human language. Recently, it has gotten increased attention from the deep learning society. Large data companies have begun investing resources into creating and developing large-scale state-of-the-art (SOTA) language models (LMs). These LMs are deep neural network architectures composed of different variants of a specific encoder-decoder architecture called the Transformer (Vaswani et al., 2017). In 2019, a team of researchers at Google AI Language released *BERT* (Devlin et al., 2019) an LM with 110 million trainable parameters[1] that achieved SOTA results across several NLP tasks. Later, in 2019, OpenAI released *GPT-2*, another LM, which also achieved SOTA results (Radford et al., 2019), but with the increased number of 1.5 billion trainable parameters. Most recently, in June 2020, OpenAI developed yet another groundbreaking LM called *GPT-3*. GPT-3 did not only achieve SOTA results across numerous NLP tasks, but it is also considered to be the largest model ever created. GPT-3 has a whopping number of 175 billion trainable parameters (Brown et al., 2020) making it ~116 times bigger than GPT-2 and ~1,590 times bigger than BERT. While most of these large-scale models achieve impressive results, Emma Strubell et al., (2019) has brought forth issues regarding the energy consumption and the environmental impacts, the training of these LMs have. According to their research, training a BERT on a GPU has the same carbon emission as a trans-American flight (Strubell et al., 2019). Other researchers have estimated that the training of a GPT-3 on a single GPU would take ~28,000 days (Anthony et al., 2020), emitting the carbon equivalent of driving a new car for 703,808.01 km, or approximately, driving a car to the moon and back. Thus, there is an issue regarding the carbon footprint and the energy consumption of developing large-scale LMs.

Another problem of the NLP community is that many of the data companies focus their resources towards the internationally spoken English language. This leaves more scarcely spoken languages, such as Danish, a language only spoken by roughly 6 million people (‘Danish Language’, 2020), in a technologically disadvantageous position. To that end, the training of these LMs requires large amounts of data (Brown et al., 2020; Radford et al., 2019; Raffel et al., 2020), and quality-assured and structured Danish text resources are limited (Kirkedal et al., 2019). However, researchers have begun collecting and structuring collections of Danish text data (Hvingelby et al., 2020; Strømberg-Derczynski et al., 2020), and when

[1] There are different versions and sizes of BERT. In this paper, 'BERT' will refer to the base version.

combining this with the environmental challenges faced with training large LMs, there is clear incentive to develop Danish energy- and sample-efficient LMs.

This paper, therefore, presents *Ælæctra*. Ælæctra is a Danish LM, created with a novel and more efficient modeling approach called *ELECTRA* (Clark et al., 2020), and pretrained on *The Danish Gigaword Corpus* (DAGW) (Strømberg-Derczynski et al., 2020). The language capabilities of Ælæctra will be evaluated on a Named Entity Recognition (NER) task and compared to two other LMs with Danish language capabilities, multilingual BERT (Devlin et al., 2019), and Danish BERT (*Danish BERT*, 2019/2020).
The paper will begin by summarizing some of the recent improvements within NLP, including how the use of transfer learning enabled more proficient models, the Transformer, BERT, and ELECTRA. A brief introduction to the relevant Danish NLP resources will be outlined and the methodology behind the pretraining and evaluation of Ælæctra will be presented. The results of the evaluation will be discussed, alongside the limitations and future improvements of both Danish and general NLP capabilities.

2.1 NLP and Transformer-based Language Models

NLP has recently seen a growth in the use and implementation of transfer learning methods. This has led to a substantial increase in the SOTA performance on several NLP tasks. The specific type of transfer learning that has enabled this increase has also been referred to as *sequence transfer learning*. Sequence transfer learning consists of two different phases: The first phase is the *pretraining* phase, where a model acquires a general representation of the general-domain of interest. The second phase is the *fine-tuning* phase where a model is trained to gain a more domain-specific representation towards a specific task. (Ruder et al., 2019). The recent improvements of NLP and the increased use of transfer learning has been heavily motivated by the introduction of the Transformer (Rogers et al., 2020).

2.1.1 The Transformer

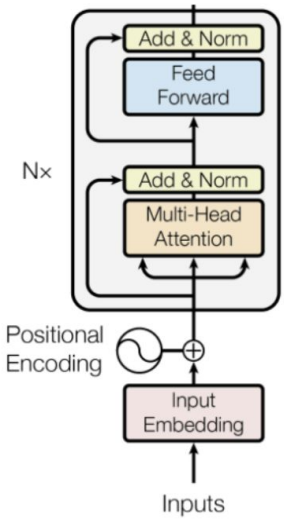
Traditionally, machine learning methods such as Recurrent Neural Networks and Long Short-Term Memory (LSTM) layers have been achieving SOTA results in regards to sequence modeling and language processing (Graves et al., 2013; Hochreiter & Schmidhuber, 1997; Sutskever et al., 2014; Vaswani et al., 2017). These methods work in a sequential manner where for each input of a sequence, the current hidden state is computed by considering the previous input and the current input position. However, this method is limited to only process

one input at a time, which makes it computationally suboptimal. This is where the Transformer is superior as it employs multi-headed attention-based mechanisms that allow for a parallelized processing of inputs (Vaswani et al., 2017). Furthermore, while traditional word-embeddings like Word2Vec (Mikolov et al., 2013) are static and incapable of capturing context, the multi-headed attention mechanisms allow the Transformer to generate contextualized embeddings. The original Transformer employs a decoder-encoder architecture, however, since the models introduced in this paper only consist of encoder layers, the focus will, solely, be put hereon. For an explanation of the decoder, see Vaswani et al. (2017).

2.1.1.1 The Encoder

A Transformer-based encoder block consists of a multi-headed self-attention layer (MHSA) and a point-wise feed-forward neural network layer (FFN), encased with residual connections and layer normalization. (Vaswani et al., 2017). The structure of the encoder can be seen in Picture 1.

When a sequence of tokens, or words, is passed through an encoder, the tokens are initially mapped onto word-embeddings, which are combined with positional encodings. Afterward, the tokens are passed through multiple self-attention heads. The way a self-attention head calculates attention is by computing the relevance of each token in a sequence relative to the other tokens in the same sequence. This is done, by calculating, for each input, a query, a key, and a value. Each query, key, and value is represented as a vector, and the scaled dot-product between these vectors is calculated to obtain the weighted representation, or the relevance, of the tokens in a sentence. MHSA refers to the fact that the above-described algorithm is applied to the tokens of a sequence multiple times, simultaneously. This allows the Transformer to attend to information from different positions of the sequence, enabling the parallelized processing of inputs. The outputs from the MHSA are, subsequently, concatenated, and run through the FFN to elicit an output vector. The outputs from the FFN are contextualized embeddings of the tokens in the input sentence and can be used as input to classification layers or additional encoders.



Picture 1: The structure of the encoder in the Transformer. Adapted from: "Attention Is All You Need" by Vaswani et al. (2017).

2.1.2 BERT

One of the well-known Transformer-based models with encoders is called *"Bidirectional Encoder Representations from Transformers"*, or BERT (Devlin et al., 2019). BERT was introduced by Devlin et al. (2019) and outformed previous SOTA models across several NLP tasks including a natural language understanding tool, GLUE (Wang et al., 2018), and a Question Answering dataset, SQuAD (Rajpurkar et al., 2016). BERT was originally an English LM, but there have been releases of BERT in other languages such as French, *CamemBERT,* and Dutch, *BERTje* (de Vries et al., 2019; Martin et al., 2020). Devlin et al. (2019) also released multilingual BERT (mBERT) pretrained on Wikipedia corpora from more than 100 languages. mBERT has been shown to generalize well across languages (Pires et al., 2019), however, might have particular deficiencies for the Danish language, and due to encoding issues, has been reported to show very little support of the Danish vowel 'Å' (Strømberg-Derczynski et al., 2020). In 2019, the Danish chatbot company, BotXO, released a Danish version of BERT (DaBERT) ('BotXO Has Trained the Most Advanced Danish BERT Model', 2019; *Danish BERT*, 2019/2020). It has been reported to achieve SOTA results in various Danish NLP tasks (*DaNLP*, 2019/2020). While mBERT is available in both a cased and an uncased version (mBERT Cased and mBERT Uncased), DaBERT only exists in an uncased version.

2.1.2.1 Pretraining BERT

The architecture of BERT consists of 12 Transformer-based encoder blocks stacked, where the hidden size of each FFN is 768, resulting in a model with 110 million trainable parameters. For a full description of the hyperparameters see Appendix A.

2.1.2.1.1 WordPiece Tokenization

When pretraining BERT, either single sentences or two-sentence couples, are passed into the encoder-stack. These sentences consist of words tokenized following a tokenization method called WordPiece tokenization (Wu et al., 2016). This method divides words into sub pieces allowing BERT to better process words outside of the BERT's vocabulary, which consists of 30,000 tokens. Furthermore, two indicator tokens, *'[CLS]'* and *'[SEP]',* are added to the beginning and ending of each sentence, respectively, letting BERT know when a sentence begins and when it ends. An example of a WordPiece tokenized sentence with indicator tokens from an Ælæctra tokenizer can be seen below:

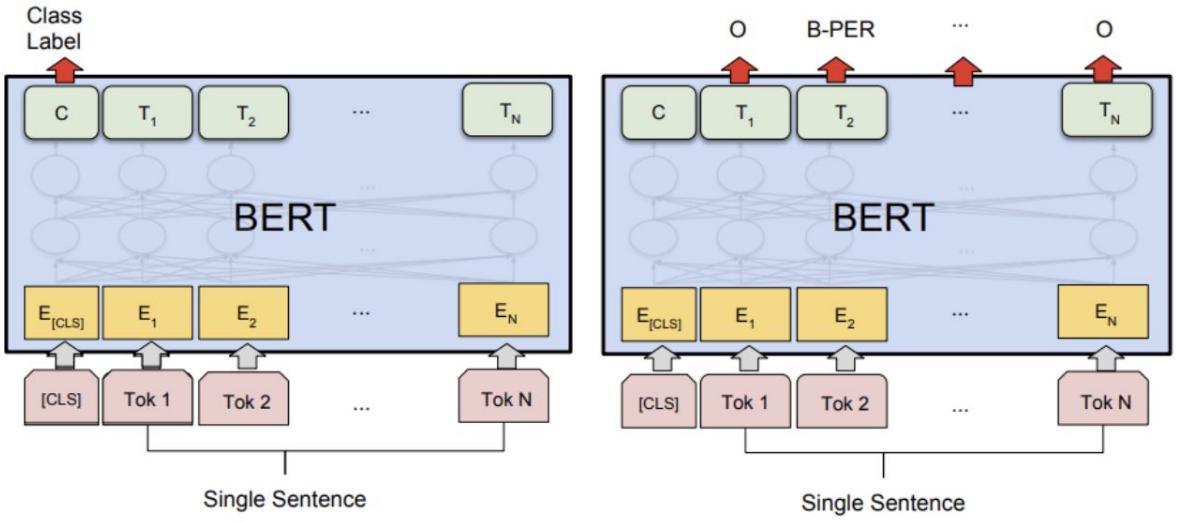
Original sentence: *"Ælæctra er en dansk sprogmodel"*
Tokenized sentence: *"[CLS] Æ ##læ ##ct ##ra er en dansk sprog ##model [SEP]"*

2.1.2.1.2 Masked Language Modelling and Next Sentence Prediction

The pretraining of BERT consists of two self-supervised tasks: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). In MLM 15% of the input tokens are masked and BERT is tasked to generate the words behind the masks. In NSP 50% of the sentence couples are randomly mixed, meaning that half of the sentences belong together, and the other half do not. Here BERT is tasked with answering if two sentences belong together or if they do not. Devlin et al. (2019) argue that NSP helps the model in question answering and natural language inference tasks, however, research has found that training a BERT without NSP matches or even improves performance (Liu et al., 2019). Therefore, it seems that MLM is the most crucial part of the two tasks.

2.1.2.2 Fine-tuning BERT

A fine-tuning of BERT is not as extensive as a pretraining. It also requires the inputs to be WordPiece tokenized, but what differentiates a fine-tuning from a pretraining is that the model weights are initialized from a pretrained BERT, and, additionally, that a final classification layer is added as an output layer to the end of the encoder-stack. If the task is token-level classification e.g., NER, the different tokens will be passed into the output layer. If the task is a single sentence classification, e.g., sentiment analysis, the *'[CLS]'* token, representing the whole sentence, will be passed into the output layer. See Picture 2 for illustrations of two examples of BERT fine-tuning.



Picture 2: Illustration of fine-tunings of BERT for token classification and single sentence classification. Adopted from: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by Devlin et al. (2019).

2.1.3 ELECTRA

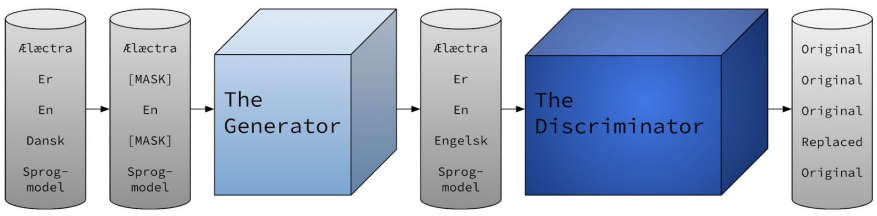
In an attempt to create a more efficient model, Clark et al. (2020) developed a more sample-efficient pretraining approach of the BERT model-structure, which resulted in the English LM *ELECTRA*, which stands for '*Efficiently Learning and Encoder that Classifies Token Replacements Accurately'*. Clark et al. (2020) showed that the small version of ELECTRA (ELECTRA-Small) was able to outperform GPT, which, in comparison, requires 30 times more computation. The base version of ELECTRA (ELECTRA-Base) was able to beat many of the SOTA LMs on both GLUE and SQuAD. While ELECTRA-Base requires the same amount of training time and computation as BERT, ELECTRA-Small can be pretrained on a single Tesla V100 GPU in four days.

2.1.3.1 Pretraining ELECTRA

The structure and the hyperparameters of ELECTRA-Base are almost identical to BERT, in that they only differ in the number of training steps, where BERT is trained for 766,000 training steps and ELECTRA for 1 million.

2.1.3.1 Replaced Token Detection

Before pretraining an ELECTRA model, all tokens are WordPiece tokenized. The pretraining method ELECTRA employs is called "*replaced token detection" (RTD).* See Picture 3 for a simplified illustration of RTD. In RTD both a *generator* and a *discriminator* are trained. The generator is tasked with MLM, where it has to generate the original tokens behind the 15% masked tokens in the input sentences. The discriminator is tasked with discriminating whether the outputs from the generator are original or replacements. This means that if the generator generates a wrong word, the discriminator has to output that the word was replaced. If the generator generates a word correctly, the discriminator has to output that the word is original. This allows the discriminator to calculate a loss across all the tokens, rather than just the masked ones. In other words, for each sample of tokens, it learns from 100% of its predictions, whereas the generator only learns from the masked tokens. This makes it much more sample-efficient, both in terms of the required amounts of data and energy, compared to the pretraining of BERT.



Picture 3: Illustration of the replaced token detection method.

After the pretraining, the generator is discarded and the discriminator becomes ELECTRA. It was discovered that a smaller generator was more beneficial and the generator for ELECTRA-Base is, therefore, ⅓ the size of the discriminator, while the difference for ELECTRA-Small is ¼. (Clark et al., 2020).

2.1.4 Ælæctra

The current performance of Danish NLP LMs and the technological development hereof is relatively inferior compared to English and other languages. Danish researchers have called for a need for improvements of Danish language technology, and accordingly, reducing the risk of only having substandard language-technology capabilities. (Kirkedal et al., 2019). As mentioned, a Danish version of BERT (DaBERT) has been released, however, the training of DaBERT was done on unstructured data that has previously shown to be of poor quality (Radford et al., 2019). Nevertheless, DaBERT achieves benchmarking results across several NLP tasks (*DaNLP*, 2019/2020). Recently, however, researchers have begun a collection of a structured and quality-assured Danish text dataset, *The Danish Gigaword corpus (DAGW)* (Strømberg-Derczynski et al., 2020). How this dataset enhances Danish NLP and the performance of Danish LMs remains unknown. The development of DAGW and the previously mentioned recent technological increments in the efficiency of training Transformer-based LMs establishes the motivation for the creation of a new efficient Danish LM. This paper, therefore, introduces *Ælæctra*. Ælæctra is both trained as a cased and an uncased version (Ælæctra Cased and Ælæctra Uncased). It is pretrained with the Danish Gigaword corpus by employing the ELECTRA-Small modeling approach. The following sections will evaluate the language capabilities of Ælæctra on two NER datasets, and compare it against DaBERT and mBERT.

3 Materials and Methods

The following section is composed of two parts. The first part presents Ælæctra and gives an overview of the pretraining of Ælæctra, including an introductory section to the data, the preprocessing of the data, and the pretraining. The second part consists of a description of the experimental setup and the evaluation process, including data preprocessing, model fine-tuning, the metrics used to assess the performance and the results from the NER tasks.

3.1 Pretraining Ælæctra

Before describing the different steps for the pretraining of Ælæctra it is of utmost importance to note that Ælæctra is the *'small'* version of ELECTRA (Clark et al., 2020). This decision was made in the interest of time and due to computational limitations, e.g, not having access to tensor processing units (TPUs). Therefore, and to reiterate: Ælæctra is a Danish version of ELECTRA-Small.

3.1.1 Data

3.1.1.1 Danish Gigaword

The data used for the pretraining of the Ælæctra is composed of a work-in-progress tokenized version of *The Danish Gigaword corpus* (DAGW) (Strømberg-Derczynski et al., 2020). It was collected from the DAGW GitHub repository[2] on the 5th of October 2020. The full size of the DAGW at the time of collection was 7.47 GB.

DAGW is a danish text corpus that was built to accommodate for the lack of work and capabilities within Danish NLP. It was constructed to create a freely available danish text corpus, that represents the Danish language, is easily applicable, and enhances the development of Danish NLP. The researchers made an active effort into only including high-quality data. (Strømberg-Derczynski et al., 2020). For example, DAGW only includes a quality-assured part of the Danish Common Crawl dataset as it has previously been shown to have quality issues, resulting in diminished model performances (Radford et al., 2019; Raffel et al., 2020). Furthermore, the Common Crawl data contains text from communities with discriminatory tendencies (racism, nazism, etc), and introducing such biases can lead to LMs eliciting toxicity and discrimination (Gehman et al., 2020; Radford et al., 2019). This might not be favorable when trying to employ *Responsible AI* (Barredo Arrieta et al., 2020). Anyhow, because of this meticulous data collection, DAGW is an advantageous dataset to use when training a Danish LM.

3.1.1.1.1 DAGW Preprocessing

The structure of DAGW is composed of many different files with lines of text, and these were needed to be collected into a single pretraining file. To do this, each sentence was tokenized into separate tokens. This was done in *Python 3.6* by using the package '*NLTK'* (Bird et al.,

[2] https://github.com/ITUnlp/dagw

2009). Afterward, the tokens were reforged into sentences, and each sentence was written as a line in the pretraining file (see Table 1 for a summary of the text dimensions in the pretraining corpus). All the code used for this project can be found on Github[3].

Total number of characters:	6,005,554,225
Total number of tokens:	974,898,201
Total number of sentences:	67,679,066
Total size:	7.47 GB

Table 1: Showing the text dimensions of The Danish Gigaword Corpus training file for Ælæctra.

3.1.1.1.2 Preprocessing for Pretraining

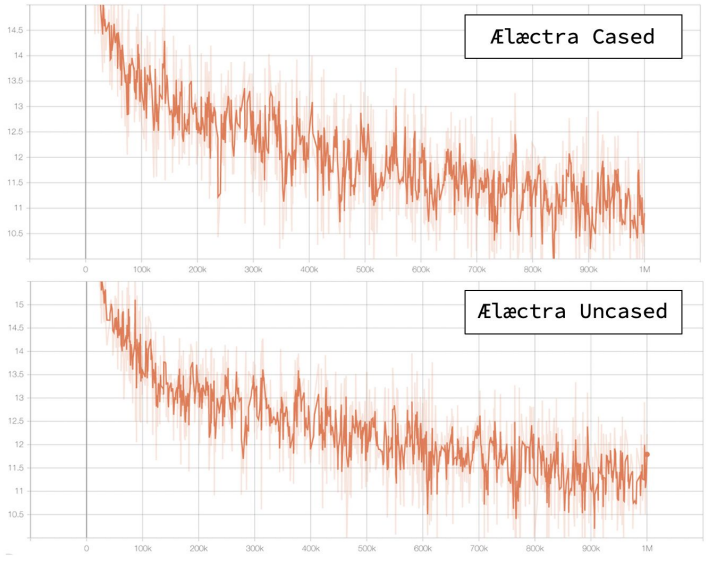
After the training file had been created, it was used to create a vocabulary file necessary for the WordPiece tokenization. This was done by using the HuggingFace (Wolf et al., 2020) package *Tokenizers*. As mentioned, mBERT has shown deficiencies when processing the letter 'Å' (Strømberg-Derczynski et al., 2020). This is due to certain types of Unicode normalizations, which causes 'Å' to be treated as an accented character. To solve this, the code was modified to not strip the tokens of accents, and resulted in a tokenizer capable of processing 'Å'. Because of the interest in both creating an *Æ*læctra capable of distinguishing between cased and uncased letters, and a model that disregards casing, both a cased and an uncased vocabulary was created. As DaBERT employed a vocabulary size of 32,000 tokens (*Danish BERT*, 2019/2020), the vocabulary size for *Æ*læctra was set to be the same to allow for comparison.

3.1.2 The Model - *Æ*læctra

The pretraining of *Æ*læctra was done by using a modified version of Google's ELECTRA script[4] in *Python 3.6*. The machine learning framework used was version 1.15 of the *Python*-package '*TensorFlow'* (Abadi et al., 2016). Before pretraining the training data file was turned into several TensorFlow native files called '*TFRecords'*. These are records of binary strings that allow for enhancement of the performance of the whole training-pipeline. The hyperparameters of *Æ*læctra are identical to ELECTRA-Small (Clark et al., 2020). For a full outline of the hyperparameters see Appendix B.

[3] https://github.com/MalteHB/-l-ctra
[4] https://github.com/google-research/electra

The pretraining was done by utilizing a single NVIDIA Tesla V100 GPU with 16 GiB, endowed by the Danish data company KMD. The pretraining took approximately 4 days and 9.5 hours for both the cased and uncased model. A loss curve for each model can be seen in Picture 4. It is not clear whether any of the models reached a global minimum. More training steps could, therefore, result in better performances, however, due to time limitations, this was not further researched.



Picture 4: Showing the loss of Ælæctra Cased and Ælæctra Uncased after 1e6 training steps.

After the pretraining of the Ælæctra models, the TensorFlow model checkpoints were converted into *'PyTorch'* model files. This allows for any future research or NLP experiments to freely choose between the two machine learning frameworks *'TensorFlow'* and *'PyTorch'*. The pretrained Ælæctra models are available from the *HuggingFace Model Hub*[5, 6].

3.2 Fine-tuning and Evaluation of Ælæctra

3.2.1 Named Entity Recognition

For the evaluation of the different models, Named Entity Recognition (NER) was chosen as the evaluation task. NER is a multilabel token-level classification task and is considered to be a crucial NLP component (Hvingelby et al., 2020; Ruder et al., 2019). English NER tools and LMs achieve high-performance metrics (F1-scores in the 90s), while for Danish it is considerably

[5] https://huggingface.co/Maltehb/-l-ctra-danish-electra-small-uncased
[6] https://huggingface.co/Maltehb/-l-ctra-danish-electra-small-cased

lower (F1-scores in the 80s) (*DaNLP*, 2019/2020; Hvingelby et al., 2020; Kirkedal et al., 2019). The purpose of NER is to classify specific entities in text, and some of the most common entities are person names, locations, organizations, time variables, and monetary values (Ruder, 2019). The way the named entities are tagged follows a tagging scheme called BIO-tagging, where the different words are separated as either being the beginning (B) of an entity, inside an entity (I), or other (O), meaning that a word is not part of the defined entities. The NER tags and their corresponding meanings, used to evaluate the models in this paper, are illustrated in Table 2.

NER tag	**Meaning**
B-PER	Beginning of person name
I-PER	Inside a person name
B-LOC	Beginning of location
I-LOC	Inside a location
B-ORG	Beginning of organization
I-ORG	Inside an organization
O	Other

Table 2: The NER tags used in this paper and their corresponding meanings.

3.2.2 Data

For the fine-tuning and evaluation of the different LMs two different datasets were used for testing: A dataset provided by KMD and a dataset, DaNE, created by Hvingelby et al. (2020). Both are tagged for named entities.

3.2.2.1 KMD Dataset

KMD is a Danish IT-company that provides software solutions, primarily, to the Danish public sector. The dataset provided by KMD is a dataset collected through the business solution *'KMD Spend Analysis'* (SA). SA is a solution that optimizes the procurement process of companies and organizations (*KMD Spend Analysis - Powered by SAS | KMD*, n.d.). The specific dataset used in this paper is an invoice dataset from the SA solution, consisting of invoice line descriptions annotated with corresponding named entities for person names and addresses. The NER tags for the KMD dataset are, therefore, *"B-PER", "I-PER", "B-LOC", "I-LOC",* and *"O"*. See Table 3 for an example of a fabricated invoice line with corresponding NER tags.

Invoice Line	*"Malte Højmark-Bertelsen bought ten bananas from the fruit store on Ripe Street No. 10."*
NER tags	*"B-PER I-PER I-PER O O O O O O O O B-LOC I-LOC I-LOC I-LOC"*

Table 3: Illustrating how an invoice line and the corresponding NER tags of the KMD SA dataset could look.

The data contains 34,147 sentences divided into training, validation, and test datasets with a split of 90%, 5%, and 5%, respectively. As the data contains personal information such as person names and addresses it will not be made publicly available. See Table 5 for the data distribution of the training/validation/test splits.

	Names (B-PER & I-PER)		Addresses (B-LOC & I-LOC)		Other Words (O)		Tokens		Sentences	
	N	%	N	%	N	%	N	%	N	%
All Data	4,184	3.1	699	0.5	131,190	96.4	136,073	100	34,147	100
Training Data	3,418	2.8	623	0.5	117,824	96.7	121,865	89.6	30,733	90
Validation Data	164	2.4	25	0.4	6,603	97.2	6,792	5	1,707	5
Test Data	602	8.1	51	0.7	6,763	91.2	7,416	5.4	1,707	5

Table 5: Showing the distribution of the KMD datasets.

3.2.2.2 DaNE Dataset

The other dataset used is the DaNE dataset created by Hvingelby et al. (2020). DaNE is a NER annotated version of the Danish Dependency Universal Dependency Treebank dataset (Johannsen et al., 2015). It contains the NER tags: *"B-PER", "I-PER", "B-LOC", "I-LOC", "B-ORG", "I-ORG", "B-MISC", "I-MISC"* and *"O"*. The *"MISC"* tag (miscellaneous) has previously been reported to have limited use, and will, thus, be excluded from this evaluation (*DaNLP*, 2019/2020; Hvingelby et al., 2020). To see the distribution of the training, validation, and test datasets of DaNE see Table 6.

	Names (B-PER & I-PER)		Addresses (B-LOC & I-LOC)		Organizations (B-ORG & I-ORG)		Other Words (O)		Tokens		Sentences	
	N	%	N	%	N	%	N	%	N	%	N	%
All Data	2741	2.7	1371	1.3	1619	1.6	95,002	94.3	100733	100	5512	100

Training Data	2146	2.6	1144	1.4	1267	1.5	75,821	94.3	80,378	89.6	4383	79.5
Validation Data	277	2.6	126	1.2	131	1.2	9,798	94.9	10322	5	564	10.3
Test Data	318	3.2	101	1	221	2.2	9,383	93.6	10023	5.4	565	10.3

Table 6: Showing the distribution of the DaNE datasets.

3.2.3 Experimental Setup

3.2.3.1 Data Preprocessing

The first step of the data preprocessing was loading the training/validation/test datasets for the two different data sources and turning these datasets into lists containing sentences with the words and the corresponding tags. The indicator tags *'[CLS]'* and *'[SEP]'* were added to each of the models' vocabularies, and subsequently, all the words were tokenized using the pretrained tokenizers from each of the corresponding models. Since WordPiece tokenization results in several words being split into separate sub pieces, the tags for these tokenized words were extended accordingly. All the sentences needed to have the same dimensions, and they were thus zero-padded (Tixier, 2018). For Ælæctra the sentences were zero-padded with the maximum input-length of the model; a maximum positional embedding length of 128. Since none of the sentences in either of the datasets exceeded the maximum positional embedding length for the BERT models (with a maximum positional embedding length of 512), it was set to be the length of the longest sentence present in the training data. Additionally, the token *'[PAD]'* was added to the vocabularies, which would be used as an attention mask indicator. The attention masks would ensure that the models only learned from the actual words and not from the added zero-paddings. The final preprocessing step consisted of taking each sentence, with the tokens and the tags, and turning this into tensors containing integers corresponding to the index number of each token derived from each of the models' vocabularies. Once this was done, the data was ready for the model fine-tuning.

3.2.3.2 Model Preparation

To prepare the models for fine-tuning to the NER tasks the *Python*-package from HuggingFace called *'Transformers'* (Wolf et al., 2020) was used. To prepare the models, a token-level classification layer was added as the final layer of the different models. This layer would receive the hidden state of each token, and assign a NER tag dependent on a softmax-calculated

probability distribution. By adding the final classification layer one can assess specific model capabilities, calculate losses through the different token prediction, and use this loss to fine-tune, through backpropagation, all of the weights and parameters of the initial model.

3.2.3.3 Parameters

After the classification layer was added, the models were initialized with the weights and different parameters created from the pretraining, resulting in the number of trainable parameters seen in Table 7. Here it is evident that the Ælæctra models are 1/8th of the size of DaBERT and at least 1/12th of the size of the mBERT models.

Model	**Trainable Parameters**
mBERT Uncased (Devlin et al., 2019)	166,771,976
mBERT Cased (Devlin et al., 2019)	177,269,000[7]
DaBERT (Danish BERT, 2019/2020)	110,032,904
Ælæctra Uncased	13,674,248
Ælæctra Cased	13,674,248

Table 7: Showing the trainable parameters of the different models.

The hyperparameters also play an important role when fine-tuning, and these were heavily inspired by the recommendations from the authors of the original BERT paper (Devlin et al., 2019). The number of epochs was chosen to be 4, as exploratory analyses suggested that more would cause the models to overfit the training data. The chosen batch size was 32, and the optimizer used was AdamW (Loshchilov & Hutter, 2019) with an initial learning rate of 3e-5. A learning rate scheduler was implemented for the models to automatically reduce the learning rate through different epochs. It is important to note that there are several other hyperparameters, and choosing different ones might elicit other results than this paper presents. For a list of the chosen hyperparameters, see Appendix C.

3.2.3.4 Metrics

To assess the model efficiency and performance the average inference time for each epoch, as well as the micro F1-score, was calculated. The calculation of the micro F1-score was done by

[7] The reason for the additional layers in mBERT Cased is because of mBERT Cased being a newer version of BERT with additional optimizations and languages.

using the *Python*-package *'SciKit-learn'* (Pedregosa et al., 2011). Present in both the KMD and the DaNE dataset is a heavy class imbalance, where e.g., more than 90% of the tokens are labeled with the *'O'*-tag. To accommodate for this the micro-averaged F1-score (micro F1) was calculated. Micro-averaging means to aggregate the contributions of all the individual NER tags and then calculate the F1-score, as opposed to macro-averaging, where you calculate the F1 for each NER tag and then take the average F1. To calculate the micro F1, firstly a calculation of the aggregated *true positives* (tp), the *false positive* (fp), and the *false negatives* (fn) was done:

$$tp_{aggregate} = \sum_{i=1}^{n} tp_i$$

$$fp_{aggregate} = \sum_{i=1}^{n} fp_i$$

$$fn_{aggregate} = \sum_{i=1}^{n} tn_i$$

Afterward, the micro-averaged precision and the micro-averaged recall was calculated, where ε was added to avoid division by zero:

$$micro\ precision = \frac{tp_{aggregate}}{tp_{aggregate} + fp_{aggregate} + \varepsilon}$$

$$micro\ recall = \frac{tp_{aggregate}}{tp_{aggregate} + fn_{aggregate} + \varepsilon}$$

Finally, by calculating the harmonic mean of the micro-averaged precision and the micro-averaged recall, the micro F1 was found, where ε again was added to avoid zero-division:

$$micro\ f1 = 2 \cdot \frac{micro\ precision \cdot micro\ recall}{micro\ precision + micro\ recall + \varepsilon}$$

3.2.3.5 Training and Evaluation process

The training process consisted of passing the training data through each model and calculating the micro F1 for the validation data predictions to assess intermediate performance. After the four epochs, the average inference time (AIT) in seconds per epoch was calculated and the training and validation loss was assessed to make sure that the models had not been exposed to a significant amount of overfitting. The models were saved and finally, each model was used to predict the testset, and from those predictions, the micro F1 was calculated.

To evaluate the different models, an iterative approach was taken, where five separate runs of the whole training/validation and test process were conducted. For each run, five different training seeds were set resulting in five different initializations of training weights. It is important to note that the creators of the DaNE dataset (Hvingelby et al., 2020) report their model performances of mBERT, (micro F1 = 83.93) and DaBERT (micro F1 = 86.61), however, due to replication issues, they will be excluded in this paper. The issue of replication is a current problem within the NLP society, where there are several reports of how different initializations of weights, different sequences of training data, and other factors, result in different performances (Crane, 2018; Dodge et al., 2020; Rogers et al., 2020). The results of this evaluation should, therefore, be interpreted with caution. The reported test results are the AIT, the micro F1 for each run, and the average micro F1 across the five runs, alongside the corresponding standard deviation (SD) for each model (for the micro F1s for each NER tag see Appendix D).

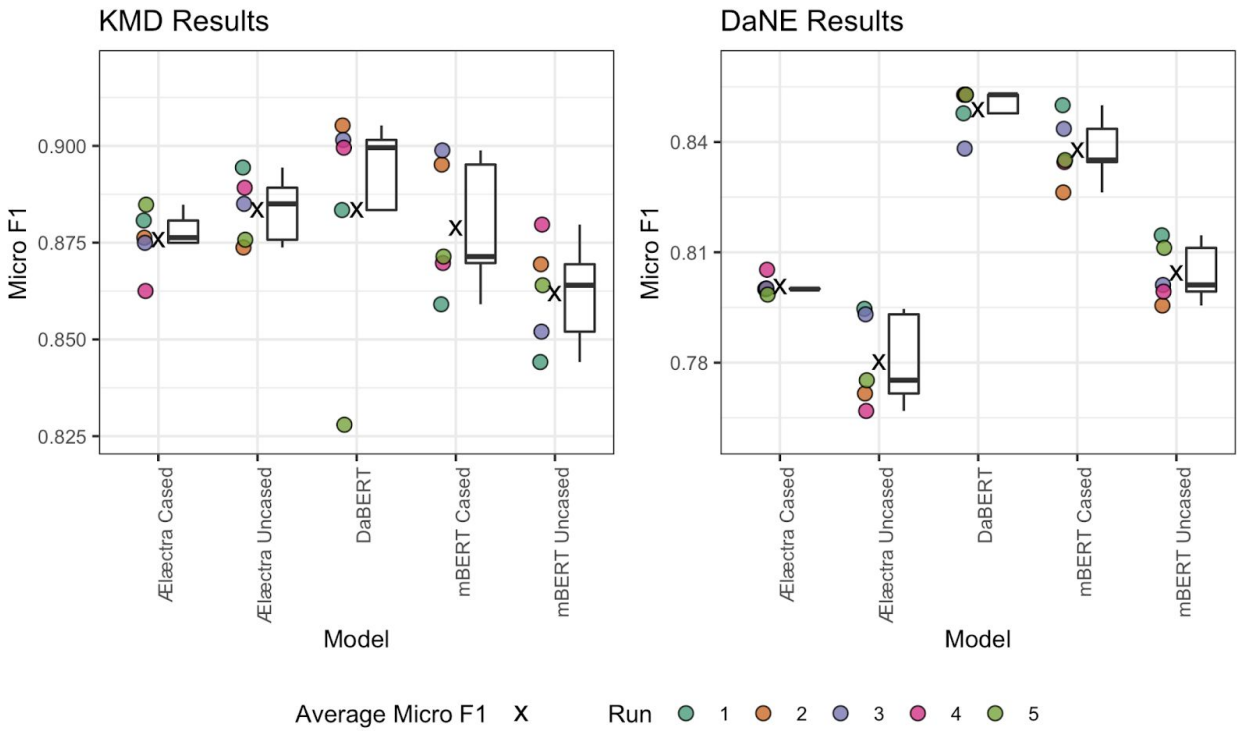
4 Results

Conducting the experimental analysis, on the two different datasets, using the iterative approach, resulted in the slew of results seen in Table 8 and illustrated in Picture 5 and Picture 6. For the KMD dataset, Ælæctra Uncased elicited the highest performance, micro F1 = 88.36 (SD = 0.88, AIT = 74.01), while the second to best model DaBERT achieved a micro F1 of 88.35 (SD = 3.22, AIT = 232.85). Ælæctra Cased achieved a micro F1 of 87.59 (SD = 0.84, AIT = 72.67). For the DaNE dataset the model with the best performance was DaBERT, micro F1 = 84.89 (SD = 0.64, AIT = 43.03), where Ælæctra Uncased achieved a micro F1 of 78.03 (SD = 1.28, AIT = 10.91) and Ælæctra Cased achieved a micro F1 of 80.08 (SD = 0.26, AIT = 10.92).

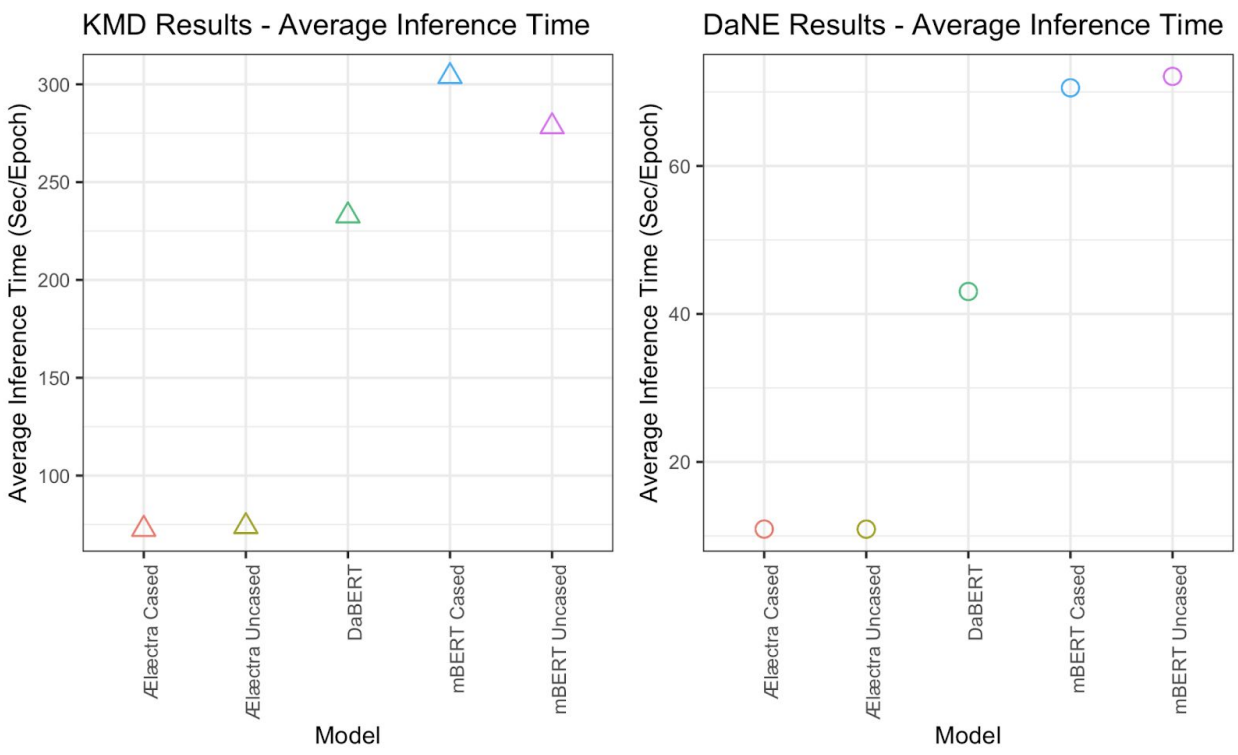
Model	**Data**	**Trainable Parameters**	**Average Micro F1**	**Average Inference Time (Sec/Epoch)**	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
mBERT Uncased	KMD	166,771,976	86.19 (SD = 1.4)	278.39	84.42	86.94	85.20	87.97	86.40
mBERT Cased	KMD	177,269,000	87.88 (SD = 1.73)	304.02	85.91	89.51	89.88	86.97	87.14
DaBERT	KMD	110,032,904	88.35 (SD = 3.22)	232.85	88.34	**90.53**	**90.15**	**89.95**	82.80

Ælæctra Uncased	KMD	13,674,248	**88.36** (SD = 0.88)	74.01	**89.44**	87.37	88.50	88.92	87.58
Ælæctra Cased	KMD	13,674,248	87.59 (SD = 0.84)	72.67	88.07	87.63	87.49	86.25	**88.48**
mBERT Uncased	DaNE	166,771,976	80.44 (SD = 0.82)	72.10	81.46	79.55	80.11	79.93	81.12
mBERT Cased	DaNE	177,269,000	83.79 (SD = 0.91)	70.56	**85.00**	82.63	**84.36**	83.45	83.51
DaBERT	DaNE	110,032,904	**84.89** (SD = 0.64)	43.03	84.78	**85.29**	83.82	**85.29**	**85.29**
Ælæctra Uncased	DaNE	13,674,248	78.03 (SD = 1.28)	10.91	79.46	77.16	79.31	76.69	77.52
Ælæctra Cased	DaNE	13,674,248	80.08 (SD = 0.26)	10.92	80.00	80.00	80.02	80.53	79.85

Table 8: Micro F1 scores for the individual runs.



Picture 5: The distribution of the micro F1s for the different runs for the KMD and DaNE datasets.



Picture 6: The average inference time (AIT) differences across all the models, and both the KMD and DaNE dataset

5 Discussion

5.1 Model Performances

5.1.1 KMD Evaluation

When evaluating the models using the KMD dataset, Ælæctra Uncased achieves the highest micro F1, but only with a 0.01 difference compared to DaBERT. Considering this small difference the results do not provide evidence of a single best LM, but rather that the performance of Ælæctra Uncased and DaBERT is indistinguishable. When further analyzing the individual runs of DaBERT, it becomes evident that the relatively high SD (SD = 3.22), is very influenced by 'run 5', micro F1 = 82.80. One could therefore speculate, that if the number of runs was increased, the results could drastically change. Looking at the other model performances it is notable, that Ælæctra Cased, micro F1 = 87.59 (SD = 0.84), performs better than mBERT Uncased, micro F1 = 86.19 (SD = 1.4), but worse than mBERT Cased, micro F1 = 87.88 (SD = 1.73). Thus, there is not a clear distinction of the language capabilities between the

*Æ*læctra models and the BERT models, however the AIT of the models differs substantially. Where the *Æ*læctra models have an AIT of around 70 seconds, the BERT models have AITs of above 230 seconds, meaning that the *Æ*læctra models are at least around 3 times faster per epoch. This clearly demonstrates the efficiency of the *Æ*læctra models, and shows how the increased sample-efficiency of the pretraining allows for the decrease in use of resources and model size while maintaining a SOTA language representation.

5.1.2 DaNE Evaluation

For the DaNE dataset the best model is DaBERT, micro F1 = 84.89 (SD = 0.64, AIT = 43.03). The second to best model is mBERT Cased, micro F1 = 83.79 (SD = 0.91, AIT = 70.56), and it thus seems that mBERT Cased outperforms mBERT Uncased, micro F1 = 80.44 (SD = 0.82, AIT = 72.10). The *Æ*læctra models perform the worst, where *Æ*læctra Cased, micro F1 = 80.08 (SD = 0.26, AIT = 10.92), exceeds *Æ*læctra Uncased, micro F1 = 78.03 (SD = 1.28, AIT = 10.91). Compared to *Æ*læctra Cased, DaBERT is 4.81 micro F1-points better. This means that across all the NER tags, DaBERT is a better classifier, but it does not show why or how DaBERT is better. An analysis of the precision and recall for each NER tag could shed light on how DaBERT is better and should be included in future evaluations. Comparing the DaNE evaluation to the evaluation on the KMD dataset, there is a decrease in SD, with the only exception being *Æ*læctra Uncased with an increase in SD (see Table 8). There is, therefore, more performance consistency across the models. This could be due to heavier data imbalances in the KMD dataset, but more investigations, with other datasets, are needed. The results from the DaNE evaluation also show how the *Æ*læctra models are much faster for each fine-tuning epoch and again demonstrates the efficiency.

In general, the results are inconclusive in that they do not elicit clearly which of the models have the best language capabilities. They do, however, show that the *Æ*læctra models are, if not similar in performance, only slightly worse, while being faster at inference time, when compared to the BERT models. Therefore, if efficiency and reduction of computational resources are priorities, *Æ*læctra could be considered to suffice as Danish LM.

5.2 Methodological Limitations

5.2.1 The iterative approach and replication issues

Given the current issues regarding replication in the NLP community (Crane, 2018; Dodge et al., 2020; Rogers et al., 2020), using the iterative approach with only five runs, might not be the best method to properly assess performance. More runs could have given a clearer and more statistically accurate depiction of the model capabilities. However, this would require more computational resources and time. There is, therefore, a need for researching alternative evaluation methods of LMs. Recently, however, researchers have started introducing other evaluation schemes, including the use of power analyses, and this could enhance the statistical evaluation capabilities and reduce the replication issues of the NLP community (Card et al., 2020).

5.2.2 Named Entity Recognition and Language Capabilities

Using NER as an evaluation task gives an idea of some of the language capabilities of an LM, but it does not elicit a complete picture. Because of NER being a token-level classification task, and because of the current experimental setup, the LMs are only tested on their ability to process information within a single sentence. Their ability to use information from previous and subsequent sentences is not investigated. Neither is their ability to perform high-level abstract language reasoning. This could be done with other NLP tasks such as Question Answering or Text Summarization, respectively. One of the reasons for only evaluating using NER, however, was due to the limited amount of Danish NLP resources, and thus, highlights the need for additional Danish NLP resources.

5.3 Future development of Ælæctra and Danish NLP

In a scientific society, such as the Danish NLP society, where there is a need for the development of better LMs, there is also a need for high-quality resources. While the introduction of DAGW is a giant step in the right direction, the largest of the four datasets of GPT-3 had the humongous size of 570 GB (Brown et al., 2020). Continuous and systematic collections of Danish text data are needed for the enhancement of Danish NLP and for the Danish language to be of technological equivalence compared to English. Furthermore, there should also be made an active effort into creating standardized Danish NLP evaluation tools.

These could be Danish adaptations of the current standard English NLP evaluation datasets, such as GLUE or SQuAD. Combining these datasets with the current research regarding evaluation of LMs could aid in the attempt to properly evaluate the language capabilities of Danish LMs. For the future development of Ælæctra, a pretraining using the base-model structure of ELECTRA, could lead to performance improvements, and thereby further enhance the Danish NLP capabilities. To sum up, it is a necessity that both the Danish and the general NLP community continues to develop NLP resources and researches both additional evaluation methods, and more efficient ways of creating high-performing and SOTA-achieving LMs.

6 Conclusion

In conclusion, this paper did not present novel SOTA results, in terms of F1-scores on NER tasks. It did, however, introduce Ælæctra, as both a cased and uncased LM, that utilizes a sample-efficient pretraining approach making it a smaller, faster, less energy-consuming, and more efficient contribution to the otherwise scarce variety of Danish LMs. In the NER tasks, when compared against current SOTA LMs with Danish language capabilities, Ælæctra achieved similar performance on the KMD dataset, but slightly worse performance on the DaNE. Therefore, if efficiency, speed and minimal energy consumption are priorities, Ælæctra might be a preferable Transformer-based Danish LM to employ. However, due to the combination of an experimental setup with an iterative approach, and current replication issues within the NLP community, caution should be displayed when interpreting practical inferences from the results. Furthermore, since NER does not succeed in eliciting a complete picture of language capabilities, Ælæctra should be evaluated on additional NLP tasks to completely establish a proper comparable evaluation of general language representation. To enhance the Danish language modeling capabilities, future work should focus on further improvements of Danish NLP resources, and in general the NLP community should continue researching efficient ways of creating SOTA LMs.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016). *TensorFlow: A System for Large-Scale Machine Learning*. 265–283. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

Anthony, L. F. W., Kanding, B., & Selvan, R. (2020). Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models. *ArXiv:2007.03051 [Cs, Eess, Stat]*. http://arxiv.org/abs/2007.03051

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc.

BotXO has trained the most advanced Danish BERT Model. (2019, December 5). *BotXO*. https://www.botxo.ai/en/blog/danish-bert-model/

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language Models are Few-Shot Learners. *ArXiv:2005.14165 [Cs]*. http://arxiv.org/abs/2005.14165

Card, D., Henderson, P., Khandelwal, U., Jia, R., Mahowald, K., & Jurafsky, D. (2020). With Little Power Comes Great Responsibility. *arXiv preprint arXiv:2010.06595*.

https://arxiv.org/abs/2010.06595

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *ArXiv:2003.10555 [Cs]*. http://arxiv.org/abs/2003.10555

Crane, M. (2018). Questionable Answers in Question Answering Research: Reproducibility and Variability of Published Results. *Transactions of the Association for Computational Linguistics*, *6*, 241–252. https://doi.org/10.1162/tacl_a_00018

Danish BERT. (2020). BotXO. https://github.com/botxo/nordic_bert (Original work published 2019)

Danish language. (2020). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Danish_language&oldid=987310297

DaNLP. (2020). [Python]. Alexandra Institute. https://github.com/alexandrainst/danlp (Original work published 2019)

de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., & Nissim, M. (2019). BERTje: A Dutch BERT Model. *ArXiv:1912.09582 [Cs]*. http://arxiv.org/abs/1912.09582

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv:1810.04805 [Cs]*. http://arxiv.org/abs/1810.04805

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., & Smith, N. (2020). Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *ArXiv:2002.06305 [Cs]*. http://arxiv.org/abs/2002.06305

Gehman, S., Gururangan, S., Sap, M., Choi, Y., & Smith, N. A. (2020). RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 3356–3369. https://www.aclweb.org/anthology/2020.findings-emnlp.301

Graves, A., Mohamed, A., & Hinton, G. (2013). Speech recognition with deep recurrent neural

networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hvingelby, R., Pauli, A. B., Barrett, M., Rosted, C., Lidegaard, L. M., & Søgaard, A. (2020). DaNE: A Named Entity Resource for Danish. *Proceedings of the 12th Language Resources and Evaluation Conference*, 4597–4604. https://www.aclweb.org/anthology/2020.lrec-1.565

Johannsen, A., Alonso, H. M., & Plank, B. (2015). Universal dependencies for danish. *International Workshop on Treebanks and Linguistic Theories (TLT14)*, 157.

Kirkedal, A., Plank, B., Derczynski, L., & Schluter, N. (2019). The Lacunae of Danish Natural Language Processing. *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, 356–362. https://www.aclweb.org/anthology/W19-6141

KMD Spend Analysis—Powered by SAS | KMD. (n.d.). Retrieved 28 October 2020, from https://www.kmd.net/solutions-and-services/solutions/kmd-spend-analysis

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv:1907.11692 [Cs]*. http://arxiv.org/abs/1907.11692

Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. *ArXiv:1711.05101 [Cs, Math]*. http://arxiv.org/abs/1711.05101

Martin, L., Muller, B., Suárez, P. J. O., Dupont, Y., Romary, L., de la Clergerie, É. V., Seddah, D., & Sagot, B. (2020). CamemBERT: A Tasty French Language Model. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7203–7219. https://doi.org/10.18653/v1/2020.acl-main.645

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *ArXiv:1301.3781 [Cs]*. http://arxiv.org/abs/1301.3781

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pires, T., Schlinger, E., & Garrette, D. (2019). How multilingual is Multilingual BERT? *ArXiv:1906.01502 [Cs]*. http://arxiv.org/abs/1906.01502

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners*. https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *ArXiv:1910.10683 [Cs, Stat]*. http://arxiv.org/abs/1910.10683

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. *ArXiv:1606.05250 [Cs]*. http://arxiv.org/abs/1606.05250

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A Primer in BERTology: What we know about how BERT works. *ArXiv:2002.12327 [Cs]*. http://arxiv.org/abs/2002.12327

Ruder, S. (2019). *Neural transfer learning for natural language processing* [PhD Thesis]. NUI Galway.

Ruder, S., Peters, M. E., Swayamdipta, S., & Wolf, T. (2019). Transfer Learning in Natural Language Processing. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, 15–18. https://doi.org/10.18653/v1/N19-5004

Strømberg-Derczynski, L., Baglini, R., Christiansen, M. H., Ciosici, M. R., Dalsgaard, J. A., Fusaroli, R., Henrichsen, P. J., Hvingelby, R., Kirkedal, A., Kjeldsen, A. S., Ladefoged,

C., Nielsen, F. Å., Petersen, M. L., Rystrøm, J. H., & Varab, D. (2020). The Danish Gigaword Project. *ArXiv:2005.03521 [Cs]*. http://arxiv.org/abs/2005.03521

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. *ArXiv:1906.02243 [Cs]*. http://arxiv.org/abs/1906.02243

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*, *27*, 3104–3112.

Tixier, A. J.-P. (2018). Notes on Deep Learning for NLP. *ArXiv:1808.09772 [Cs]*. http://arxiv.org/abs/1808.09772

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 5998–6008). Curran Associates, Inc. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2018). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 353–355. https://doi.org/10.18653/v1/W18-5446

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., & Brew, J. (2020). HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv:1910.03771 [Cs]*. http://arxiv.org/abs/1910.03771

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., … Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv:1609.08144 [Cs]*. http://arxiv.org/abs/1609.08144

Appendix

A Hyperparameters for Pretraining BERT

Hyperparameters	**BERT/ELECTRA-Base**
Number of Layers	12
Hidden Size	768
Embedding Size	768
Learning Rate	2e-4
Training Batch Size	256
Training Steps	766,000
Feed-Forward Neural Network Inner Hidden Size	3072
Attention Heads	12
Attention Head Size	64
Generator Size (Only ELECTRA-Base)	1/3
Mask Percent	15
Learning Rate Decay	Linear
Warmup Steps	10,000
Adam Epsilon	1e-6
Adam Beta 1	0.9
Adam Beta 2	0.999
Attention Dropout	0.1
Dropout	0.1
Weight Decay	0.01

B Hyperparameters for Pretraining Ælæctra

Hyperparameters	**Ælæctra/ELECTRA-Small**
Number of Layers	12
Hidden Size	256
Embedding Size	128
Learning Rate	5e-4
Training Batch Size	128
Training Steps	1e6
Feed-Forward Neural Network Inner Hidden Size	1024
Attention Heads	4
Attention Head Size	64
Generator Size	1/4
Mask Percent	15
Learning Rate Decay	Linear
Warmup Steps	10,000
Adam Epsilon	1e-6
Adam Beta 1	0.9
Adam Beta 2	0.999
Attention Dropout	0.1
Dropout	0.1
Weight Decay	0.01

C Hyperparameters for Fine-tuning the models

Hyperparameters	**Ælæctra/mBERT/DaBERT**

Fine-tuning	
Epochs	4
Optimizer	AdamW
Learning Rate	3e-5

D Average Micro F1s for each NER tag

Model	**Trainable Parameters**	**Data**	**Average Micro F1**	**B-PER**	**I-PER**	**B-LOC**	**I-LOC**	**B-ORG**	**I-ORG**
mBERT Uncased	166,771,976	KMD	86.19 (SD = 1.4)	87.2 (SD = 2.17)	88.2 (SD = 1.64)	67.8 (SD = 4.6)	70 (SD = 3.24)	-	-
mBERT Cased	177,269,000	KMD	87.88 (SD = 1.73)	87.2 (SD = 1.92)	90.4 (SD = 1.52)	70.8 (SD = 7.46)	**74** (SD = 5.83)	-	-
DaBERT	110,032,904	KMD	88.35 (SD = 3.22)	**91.2** (SD = 0.84)	89.2 (SD = 5.22)	**77.8** (SD = 4.55)	69.8 (SD = 2.49)	-	-
Ælæctra Uncased	13,674,248	KMD	**88.36** (SD = 0.88)	89.4 (SD = 0.89)	**90.8** (SD = 0.84)	67 (SD = 4.12)	59 (SD = 4.74)	-	-
Ælæctra Cased	13,674,248	KMD	87.59 (SD = 0.84)	89.6 (SD = 0.55)	90.2 (SD = 0.84)	63 (SD = 3)	54.4 (SD = 3.21)	-	-
mBERT Uncased	166,771,976	DaNE	80.44 (SD = 0.82)	90.2 (SD = 0.84)	96.8 (SD = 0.45)	74.6 (SD = 1.52)	32.2 (SD = 9.12)	70.6 (SD = 2.88)	61.4 (SD = 6.02)
mBERT Cased	177,269,000	DaNE	83.79 (SD = 0.91)	90.6 (SD = 1.67)	**98.6** (SD = 0.55)	81.2 (SD = 1.3)	32.6 (SD = 7.5)	**76** (SD = 3.16)	69.8 (SD = 2.77)
DaBERT	110,032,904	DaNE	**84.89** (SD = 0.64)	**91.6** (SD = 0.55)	98 (SD = 0)	**84.2** (SD = 0.45)	34.4 (SD = 0.89)	**76** (SD = 2.83)	65 (SD = 2.83)
Ælæctra Uncased	13,674,248	DaNE	78.03 (SD = 1.28)	81.6 (SD = 2.07)	95 (SD = 0.71)	78.6 (SD = 2.97)	32.8 (SD = 36.01)	58 (SD = 1.22)	71 (SD = 2.74)
Ælæctra Cased	13,674,248	DaNE	80.08 (SD = 0.26)	82.6 (SD = 1.52)	94.2 (SD = 0.45)	79.6 (SD = 0.55)	**61.2** (SD = 12.68)	66.6 (SD = 0.55)	**77.4** (SD = 1.34)