DaLUKE: The Knowledge-enhanced, Danish Language Model

Technical University of Denmark

Bachelor's Project in Artificial Intelligence and Data

Søren Winkel Holm Asger Laurits Schultz

s183911@dtu.dk s183912@dtu.dk

Supervisors:
Lars Kai Hansen, DTU Compute
Michael Riis Andersen, DTU Compute
Victor Elkjær Birk, IBM Services

June 28, 2021

Abstract

The advent of deep learning has led to significant advances in the field of natural language processing in recent years, but many models, while good at modeling language, lack explicit knowledge, making tasks involving real-world, factual entities challenging. LUKE, proposed by Yamada et al. in October 2020, is a transformer-based architecture that explicitly models entities, allowing it to achieve state of the art on several benchmarks, including named entity recognition (NER). In this report, LUKE's English NER results are reproduced along with leading Danish NER results, and an open source Danish LUKE, DaLUKE, is produced. Firstly, a general, pretrained model for producing contextualized word and entity representations is released. Secondly, a model is presented which is trained on the central Danish NER dataset, DaNE, achieving close to state of the art and slightly outperforming BotXO's Danish BERT, though within margin or error. Several ablation studies are conducted to explore what effects different techniques have on performance. Finally, an open source software package, `daluke`, is released with the goal of making knowledge-based deep learning for Danish easy to use.

Preface

This project has been completed to fulfil the requirements to obtain the bachelor's degree in Artificial Intelligence and Data at the Technical University of Denmark. Per author, the project was assigned a workload of 15 ECTS corresponding to approximately 420 working hours. The software presented in the project is available at `github.com/peleiden/daluke`.

We are incredibly grateful for the large amount of help we received from DTU and broadly from Danish natural language processing practitioners. Obligatory mentions include Rasmus Arpe and Martin Nielsen from Danspeech, Finn Årup Nielsen from DTU Compute, and Kasper Lindskou and Lukas Nielsen from Ekstra Bladet Analyse for valuable advice, especially on data concerns. We are also grateful for the help of Ikuya Yamada, Studio Ousia for publishing LUKE and openly answering all our questions on modeling details.

A special thank you goes out to Johannes Kruse for continuous feedback and insight. All of our three supervisors gave us invaluable guidance and encouragement crucial for the project. We would never have learned so much about this exciting field without the ambition and motivation of Lars Kai Hansen.

Finally, we note that this project presents text examples in Danish that are not translated to English. A report meant for a broader audience would require such translations.

Abbreviations

AI	Artificial intelligence
AMP	Automatic mixed precision
BPE	Byte-pair encoding
CER	Contextualized entity representation
CRF	Conditional random field
CWR	Contextualized word representation
FN	False negative
FP	False positive
GLUE	General Language Understanding Evaluation
IOB	Inside-outside-beginning
KB	Knowledge base
KG	Knowledge graph
LM	Language model
MLM	Masked language model(ing)
NE	Named entity
NER	Named entity recognition
NLP	Natural language processing
PCA	Principal component analysis
RNG	Random number generation
SB	Sub-batch size
SOTA	State of the art
STLR	Slanted triangle learning rate
TN	True negative
TP	True positive
UD-DDT	Universal Dependencies of Danish Dependency Treebank

Contents

1. Introduction

The rapid development of artificial intelligence (AI) seen in the last decade is not just one of research. The applied field sometimes called data science or AI engineering has brought emerging intelligent technologies to industry, resulting in AI playing a significant role in society. This data-driven wave of AI has been led by the subfield that deals with statistical learning such as deep neural networks, while symbolic methods using explicit knowledge have been less influential [LBH15, p. 1]. The successful societal impact of these statistical methods that include deep learning is, however, conditioned on the availability of big data and substantial computing resources, resulting in high resource domains receiving most of the benefit of AI.

A clear example of this issue is the important field of natural language processing (NLP) that attacks the difficult task of understanding human language, one of the distinguishing aspects of our intelligence. NLP based on statistical methods from AI has in recent years taken sizable strides towards this understanding as large-scale models with names such as GPT-2, GPT-3, BERT, RoBERTa, ELMo, and T5 have gained public attention and being put to use. The data hunger of these models results in languages with low amounts of available data, dubbed low resource languages, lacking severely behind in technical results and thus practical use of NLP. To avoid these regions falling behind in the AI race, these languages, that include Danish, have seen a growing trend of attention from the literature resulting in a diverse set of methods attempting to learn in the data sparse case [Hed+21].

One such possible mitigation is to let the statistical approach be influenced by explicit modeling of knowledge. In this project, language understanding in Danish will be attempted using a new deep learning technique for NLP that succeeded with this modeling approach in English. The model, *Language Understanding Using Knowledge Embeddings* (LUKE), was released in October of 2020 and explicitly handles the difference between single words and *named entities* such as "Lionel Messi", "The United States of America" and "Science" [Yam+20]. The model had in total seen training from 160 GB of text[1.1] and achieved state of the art (SOTA) results on classic NLP

tasks such as named entity recognition (NER). For Danish, the amount of suitable data for this technique is about 40 times smaller[1.2]. It is the goal of our project to present a maximally performant and open source Danish LUKE for use of Danish NLP practitioners while also taking this challenge as a case study of deep natural language processing for low resource languages.

1.1 The Project

Apart from the practical task of releasing a usable, open source Danish LUKE model (DaLUKE), the project seeks to answer the following research questions:

1. Can both the English LUKE NER result and previous Danish NER results be reproduced, and how does the DaLUKE performance and predictions compare to this Danish SOTA?
2. Do the new knowledge-based methods proposed by the LUKE paper raise the level of natural language understanding of the used Danish model?
3. What methods in the applied data engineering pipeline and the knowledge-enhanced pretraining task are important for the final DaLUKE results?

In the following chapter, the central task of NER will be introduced, and related models within both Danish and general, knowledge-enhanced NLP will be discussed, before introducing LUKE. In Chapter 3, the used datasets and our augmentation of these are introduced. Then, Chapter 4 documents the actions taken to pretrain and finetune DaLUKE and to reproduce existing results. Here, the released software package `daluke` will also be introduced. The NER results of DaLUKE and reproduced Danish models are shown in Chapter 5 along with DaLUKE pretraining results. In Chapter 6, approaches for understanding the results are shown, including ablation studies on the training method and analysis of model predictions. Finally, project outcomes are summarized.

[1.1] This includes all training that affected the final model weights and thus also the data used to pretrain RoBERTa [Liu+19, Sec. 3.2], the weights of which were used for initializing most of the LUKE weights. Counting only the data used for LUKE itself, the number is not directly reported but was found to be around 7 GB from our experiments with the English LUKE data pipeline.

[1.2] The pretraining method uses the special format of Wikipedia. The English Wikipedia dump used by LUKE contained 3.5 billion words [Yam+20, App. A], while the Danish Wikipedia consists of 81 million words.

2. Theory and State of the Art

With the advent of deep learning in the field of NLP, models have risen massively in complexity, requiring ever more compute and data. For instance, LUKE required 30 days of training on 16 NVIDIA V100's [Yam+20], and that model already took advantage of RoBERTa, which was even more demanding to train [Liu+19]. For this reason, the training of a deep NLP model is typically divided into two separate tasks: First, the *pretraining*, in which a model is trained that has limited direct practical use but has general language understanding. Secondly, a *fine-tuning*, or downstream training, is performed in which the pretrained model undergoes further, specialized training for some specific language task, e.g. NER.

2.1 Named Entity Recognition

NLP has the broad goal of algorithmically understanding languages such as Danish or English. To quantify this abstract ideal, understanding is often measured in performance on a set of well-defined challenges that are expected to require general language knowledge. One such challenge is the task of recognizing *named entities* (NE), real-world objects with rigid definitions such as specific persons, locations, organizations, events, products, etc. The problem of NER is to produce an algorithm that can, given a string of characters designating a document of natural language, return an *annotation* that ascribes, to each word in the document, both whether the word is a part of a NE, and what category of NE the word corresponds to [Wik21]. For the string

"Caesar marched on Rome, defying the Senate of the Roman Republic.",

a correct result could be

- *Caesar* is a person,
- *Rome* is a location,
- *the Senate of the Roman Republic* is an organization,
- and all other words are not part of named entities.

There are multiple ambiguities in this formulation of the problem, but this is a common issue for the problem, as the set of NE types varies between benchmarks. Furthermore, the very definition of a NE is unsettled with possible answers including proper nouns, rigid designators, and unique identifiers [Mar+13, Sec. 4]. For the benchmarks used in this project, the NE categories and annotation formats are introduced in Section 3.2 while the evaluation of the performance of NER algorithms is discussed in Section 4.1.1.

NER is not just used as a benchmark for language understanding: In the applied field of information retrieval, semantic annotation and question answering systems rely on NER to control the information focus [Mar+13, Sec. 2]. This task of both research and practical interest was defined in the 1990s and has been active and competitive, especially in English where SOTA methods achieve close to human performance on some classic benchmarks [Wik21; Mar+13]. Researchers argue, however, that the task is not yet solved as statistical algorithms fitted to a certain NER dataset often generalize poorly to new examples [Mar+13, Sec. 7.2].

The SOTA on English NER benchmarks moves quickly, but almost all algorithms currently (June 2021) achieving high scores in common benchmarks are deep neural networks and of the transformer architecture in particular [Rud21; Con21].

2.2 Embeddings, Tokenization, and the Transformer

Word embeddings Word embeddings are one of the most ubiquitous tools in the NLP toolbox and come in many variants, all sharing the general idea of mapping a word token to a real-valued vector representation corresponding to a semantic condensation of the word. When using the term *embeddings*, we refer to the static, un-contextualized representations such as the fasttext-based, 300-dimensional, Danish embeddings released by Edouard Grave et al. [Gra+18].

One of the challenges in language modeling is that many words change their meaning completely depending on the context in which they appear. Consider, for instance, the sentences "The mouse and the elephant" and "Please left-click using the mouse". "mouse" appears identically in both sentences, but with widely different meanings, which static embeddings have a hard time modeling. In the latent space of the embeddings, one would expect "mouse" to be close to other species from the animal kingdom, but also close to computer parts. This would in turn cause computer parts to be placed close to animals, which is not ideal. This issue with embeddings is where the transformer comes in.

The Transformer Introduced by Ashish Vaswani et al. in 2017 [Vas+17], the goal of the transformer is to produce *contextualized word representations* (CWR) given static but learned embeddings. CWRs are similar to embeddings in that they are real-valued vector representations of words, but they also take the document context into account.

Transformers are not the first attempt to solve this problem of producing effective CWRs. Such general-purpose, pretrained models producing CWRs are called *contextualized word representation language models* or *general-purpose language models* [Bir20, Ch. 2] and will in this project be referred to simply as *language models* (LM). Older architectures such as recurrent neural networks [GBC16, Ch. 10] and in particular long short-term memory networks [HS97] have yielded strong results, but we argue that they have three key weaknesses: Firstly, they only include context up to the given word, and secondly, they struggle with long-term dependencies due to their sequential nature resulting in poor information preservation [GBC16]. Finally, the sequentiality of their calculations also limits how parallelized training and inference can be. The first issue has been solved by the introduction of bi-directional versions of these network types [SP97]. The second issue of long-term dependencies and the third of parallelizability is what the transformer seeks to tackle.

Both of these issues are solved by the attention mechanism. Instead of sequentially letting each token update the state, the relevance of all tokens to all other tokens are calculated simultaneously, which can be done in parallel without losing any information to an iterative state update. This is exemplified at figure 2.1, right.

The transformer uses an encoder-decoder structure with the encoder producing the CWRs. The encoder is made of several attention blocks feeding into each other with feed forward networks and layer normalizations in between. Each of the attention blocks is made up of three learnable components: the query, key, and value matrices. The word embeddings are structured in a single matrix with token embeddings in the rows and are multiplied with the three matrices before the attention scores are calculated. This allows every token to be considered simultaneously, independently, and equally, solving both the described issues. [Vas+17]

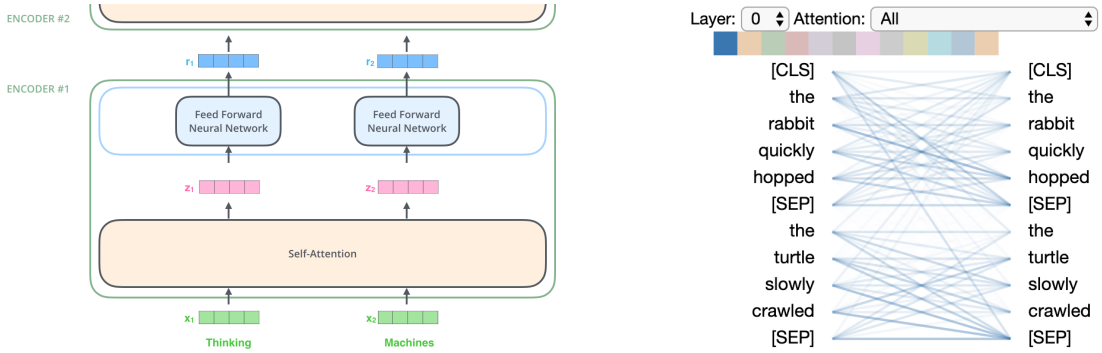

Figure 2.1: On the left, a simple visualization of an attention block, a single component computing representations, is shown. The right image exemplifies attention between all words as the BERT encoder computes it.[2.2]

Models based directly on the transformer have been trained on very large corpora to produce English language models such as BERT (*Bidirectional Encoder Representations from Transformers*) [Dev+19], RoBERTa, [Liu+19] and GPT-3 [Bro+20] that have achieved SOTA in several downstream language tasks. BERT is the first model to use the masking pretraining objective as it is employed by LUKE. Early transformer-based models such as OpenAI's GPT [Rad+18] use a unidirectional pretraining task, letting only tokens appearing to the left of the token to predict have any impact. BERT instead masks some tokens in the given sequence and allows all tokens to attend to all other tokens when predicting the masked ones – this is what is known as a masked language model (MLM). BERT achieved SOTA many downstream tasks such as the important question answering task and has been described as revolutionary for its impact on NLP [Raj19]. RoBERTa is a newer pretraining using the same architecture as BERT [Liu+19].

Tokenization The input for these language models must naturally be sequences of words. A naïve approach is to regard all unique words seen in the corpus as tokens. However, the fact that multiple possible augmentations of the same word can be met, makes this a suboptimal method. A verb, for example, still carries much of the same

[2.2]The left image is produced by Jay Alammar in "The Illustrated Transformer", `jalammar.github.io/illustrated-transformer/`. The right is a product of the Tensor2Tensor visualization tool produced by Llion Jones and used for BERT by Jesse Vig in "Deconstructing BERT", `https://towardsdatascience.com/deconstructing-bert-part-2-visualizing-the-inner-workings-of-attention-60a16d86b5c1`. Both visited 27/6 2021.

meaning between conjugations, but the naïve solution would require learning an embedding for each form. While this might be solved by lemmatizing words to their stems, this introduces the problem of information loss.

A more effective way of tokenizing has been presented by Sennrich et al. [SHB16] that uses the byte-pair encoding (BPE) compression algorithm. This method produces a token vocabulary of a given size from the corpus where each token is either a full word or a subword. BERT uses the WordPiece BPE token vocabulary which contains 30,000 tokens that have been learned with heuristic tokenization rules [Wu+16; Dev+19]. Radford et al. [Rad+19] modify BPE tokenization for the GPT-2 transformer such that splits are made on the byte level rather than on the character level, which works better on corpora containing many non-ASCII characters. This method is used by RoBERTa to produce a token vocabulary of size 50,000 [Liu+19]. LUKE directly uses RoBERTa's tokenizer when dealing with words [Yam+20].

2.3 Deep Natural Language Processing in Danish

Danish is, compared to English, a low resource language: The world's Danish-speaking population is smaller than the English-speaking population of some large cities, naturally resulting in significantly fewer and smaller available text datasets, and in less attention from both researchers and practitioners. While pushing the performance of current NLP algorithms to their limits is much more difficult in low resource languages, the field of applied NLP in Danish has adopted many of the deep learning developments of recent years.

Word embeddings are used as a core component of many NLP pipelines. In the large, open source NLP frameworks *SpaCy* [Hon+20] and *Flair* [Akb+19], which also release pretrained Danish models, the embeddings interact with different, ever-improving language models. This has successfully resulted in SOTA on multiple Danish NLP tasks. The Alexandra Institute, a Denmark based AI research and development organization, release versions of both Flair and SpaCy fine-tuned on multiple benchmarks, including NER, and report close to SOTA performance using these. The benchmarks, results, and fine-tuned Danish models are released as a part of their open source DaNLP project gathering resources for Danish NLP [Bro+21]. Another application of existing NLP frameworks was performed by Kenneth Enevoldsen, Center of Humanities Computing, Aarhus University in releasing the model *DaCy*, based on version 3 of SpaCy [Ene21].

In Danish, the most significant, pretrained language model is in our estimation a

Danish version of BERT, released by the company BotXO in 2019 [Bot19]. This model, in the following called *da-BERT*, is produced by pretraining the base architecture of BERT [Dev+19] on 10 GB of Danish text predominantly acquired from web scrapes. The da-BERT model has been fine-tuned on most Danish benchmarks in the DaNLP project and consistently yields top results [Bro+21; Hvi+20]. The model and its tokenizer are trained on lowercased text.

Another, more recent, Danish transformer is the adaption of the more resource-efficient language model Electra to Danish under the name Ælæctra by Malte Højmark-Bertelsen, KMD and Aarhus University [Høj20]. NER fine-tuned versions of Ælæctra and some multilingual transformers have been released by Ekstra Bladet Analyse under the project *NER for Danish* (NERDA) [KN20].

Of this plethora of Danish language models and NLP frameworks, the current (June 2021) SOTA on central NER benchmarks is, according to DaNLP, the 2021 newcomer DaCy. This model surpassed DaNLP's fine-tuned da-BERT model which was SOTA when we started this project[2.3]. These results are reproduced and compared with DaLUKE in this project. The NER benchmark will be presented in Section 3.2.1, the reproduction method in Section 4.1.3, and the results in Section 5.3.

Finally, the NLP scene of a language is much more than the public, pretrained models: The available rule-based tools, corpora, and other language resources are instrumental in developing practical NLP pipelines. One pipeline supplying multiple of these resources is the IT University of Copenhagen project DKIE, also including a NER model [DFB14]. A 2019 paper by NLP researchers at the same university surveyed the availability of such tools and highlight a lack of practical resources and datasets [Kir+19]. Other overviews of Danish NLP resources include those published by the Alexandra Institute at DaNLP [Bro+21] and the comprehensive list maintained by Finn Årup Nielsen, DTU Compute [Nie21].

2.4 Deep, Knowledge-enhanced NLP

The combination of statistical methods and explicit human-crafted domain knowledge was SOTA in many NLP tasks in the 1990s and up into the 2000s [RN09, Sec. 22.5]. As with most other fields of AI, the emergence of performant deep learning methods disrupted NLP in the 2010s, popularizing the methodology of using deep LMs [OMK18].

[2.3] A table of the results can be seen at the DaNLP repository at `https://github.com/alexandrainst/danlp` under `docs/docs/tasks/ner.md`.

This approach rarely includes any modeling of knowledge that is not induced implicitly by the contextual representations generated during pretraining on an unannotated text corpus.

The Defense Advanced Research Projects Agency (DARPA) identify the future wave of AI as one of *contextual adaption*; a combination of deep learning for perception and latent representations and symbolic modeling methods [Lau17]. In NLP, knowledge-enhancing deep neural networks is not just an idea for the future: Several approaches using explicit knowledge to improve pretrained contextual word representations (CWRs) have been presented in recent years, a subset of which will be summarized here.

2.4.1 Static, Separate Knowledge Graphs

A direct way to use explicit knowledge is to maintain a separate representation of facts which can be incorporated into both pretraining and inference of deep language models. This representation is often called a knowledge base (KB) – or knowledge graph (KG) if relational facts are modeled.

In 2019, *Enhanced Language Representation with Informative Entities*, ERNIE, was introduced by a Beijing team [Zha+19] looking to "enhance language representation with external knowledge" [Zha+19, p. 1] by recognizing mentions of NEs in given text and retrieving their positions in a separate knowledge graph. The query from the knowledge graph is encoded into knowledge embeddings which are taken as input for a BERT-based dual transformer architecture. The model was during training required to fill in randomly masked NEs entities in given sequences using the KG. ERNIE was not evaluated on NER the original paper, but was evaluated on the *General Language Understanding Evaluation* (GLUE) benchmark [Wan+18], where it did not outperform BERT.

Later in 2019, a similar idea was proposed in *Knowledge Enhanced Contextual Word Representations* where the model *KnowBert* [Pet+19] was shown to outperform ERNIE and BERT in knowledge related tasks, though GLUE and NER were not tested. Here, the key addition was the Knowledge Attention and Recontextualization (KAR) component which allows information to be retrieved from multiple KG's of different forms such as WordNET and Wikipedia and represented within the BERT encoder.

In KnowBert, though the KAR weights were trained, the KG was only used for encoding fixed facts for the model. The *Knowledge Graph Language Model* (KGLM) presented in *Barack's Wife Hillary: Using Knowledge Graphs for Fact-Aware Language*

Modeling [Log+19] employs a more dynamic use of the knowledge graph. This model uses the same approach to represent knowledge graph facts as ERNIE, but also builds a local graph on the sequence level which is grown by a generative model with each new token. The paper presents improved fact completion compared to GPT-2 and further analysis has shown that the overall fact completion of KGLM is comparative to KnowBert, though they perform very differently from domain to domain, with KGLM being most dependent on the used knowledge graph [Bir20].

2.4.2 Pretraining Augmentations

A strength of the statistical, deep approach that implicitly models knowledge is the potential for generalizable results: You do not need a new domain specific KG for every application if enough general knowledge has been caught in your weights. Motivated by the hope of this elusive generalizability, a number of methods have been proposed to enhance specifically the *pretraining* of language models to better handle factual knowledge, without using explicit KGs in the inference. LUKE, the subject of this project, is one of these attempts.

An example of this weaker knowledge modeling is found in the 2019 model *KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation* [Wan+21] where downstream inference can be performed without any additional resources other than the transformer structure itself. The knowledge enhancement of this BERT-based LM is, however, in the pretraining where the model optimizes over a joint task of both performing MLM and a novel knowledge graph objective requiring the model to learn knowledge embeddings. KEPLER does not improve GLUE performance compared to RoBERTa [Liu+19] from which it was initialized, but beats both classical language models and knowledge enhanced ones such as ERNIE and KnowBERT on a number of knowledge related tasks.

This idea of pretraining a transformer both for MLM and for a new knowledge-guided task has been taken up multiple times, including in LUKE. *WKLM*, presented in 2019 in *Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model* [Xio+19], uses the English Wikipedia, taking hyperlinks to be entities. The authors train an unaltered BERT architecture jointly on MLM and on a task involving identifying the randomly replaced entity in a sequence. *KALM*, published after LUKE in *Knowledge-Aware Language Model Pretraining* [Ros+20], also keeps the core language model architecture, in this case GPT-2, unchanged, adding a separate entity

tokenizer and entity embedding layer for an entity prediction pretraining task.

These models achieve good performance; WKLM surpasses BERT and ERNIE on some question answering and entity related benchmarks, and KALM outperforms GPT-2 on some knowledge related zero-shot tasks. However, the 2020 article *K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters* [Wan+20] raises a concern with the approach. According to the authors, the idea of knowledge-augmenting the pretraining task and then updating the entire model limits the ability to inject versatile knowledge and sets up a heterogeneous learning problem resulting in entangled parameters that are hard to investigate. The model *K-Adapter* instead uses "adapters", add-ons for the transformer structure, which are pretrained for different knowledge tasks while the base RoBERTa parameters are not updated. The resulting model outperforms BERT, WKLM, ERNIE, KEPLER, and KnowBERT on central knowledge related tasks.

Another issue, raised more recently, with this jungle of knowledge-enhanced models is that they do not unify generative and discriminative tasks, limiting their generalizability to fewer downstream tasks. This is voiced in *KgPLM: Knowledge-guided Language Model Pretraining via Generative and Discriminative Learning* [He+20] where the model KgPLM, pretrained by both generating masked entities and discriminating replaced ones, is proposed as a solution.

Which of the knowledge-enhanced models results in the best general performance for practitioners is beyond the scope of this project, as the continuous improvement of many of the models and large amount of different benchmarks used in the articles make direct comparisons of general language understanding ability challenging. On all benchmarks tested by the LUKE team [Yam+20], none of the above models have presented higher scores than LUKE [Yam+20, Sec. 4].

2.5 LUKE

LUKE, Language Understanding with Knowledge-based Embeddings, is a language model introduced by Yamada et al. in October 2020 [Yam+20]. As the model is applied in this project, key properties of the architecture and pretraining methodology are presented in this section.

Like other new models, LUKE builds on the BERT transformer architecture [Dev+19], adding a knowledge-enhancing pretraining task. But the LUKE encoder does not only produce CWRs; *entities* are also represented by the encoder by mapping each entity to

the same latent space as the words. In the pretraining of LUKE, entities correspond to Wikipedia articles.

Yamada et al. show that this approach leads to SOTA results on a number of entity-related tasks, such as NER, and argue that the LUKE approach mitigates the following issues with existing methods:

- Existing CWRs cannot represent spans of multiple words and thus require additional downstream modeling for language tasks in which word spans are important.
- Transformers can capture the many intricate relations between single words, but language often requires reasoning about the relationships of entities consisting of multiple words [Yam+20]. This is difficult for language models that were trained to predict "Rings" in "We watched Peter Jackson's movie, The Lord of the [MASK]", which requires less knowledge than predicting the entire "The Lord of the Rings" entity.

The LUKE idea is to consider entities as first-class citizens of text documents along with words, maintaining vocabularies of both entity and word tokens.

2.5.1 Architecture

Input Tokenization An input example for LUKE consists both of a sequence of words and a set of entities mentioned in the document, such as:

- "Biden is oldest president of the United States to date."
- {*Joe Biden* at word #1, *The United States of America* at words #7-8}.

For use of LUKE, the text sequence is tokenized into m subword tokens using the normal tokenizer of RoBERTa, the transformer on which LUKE is based. The entities are converted to n entity tokens; integer values corresponding to indices in a vocabulary of known, named entities. The positions of the entities are included in the further inference as *position IDs*.

Embeddings The subword tokens are passed through the same, static word embedding architecture as that of BERT which outputs m vectors of dimension H, one for each subword. Entity embeddings are somewhat more complex than a simple lookup table. Entities given to the model also include their positions in the sequence. The positions and the entities themselves have respective, static embeddings, with the position

embeddings residing in $\mathbb{R}^H$ and the entity embeddings in $\mathbb{R}^h$ where $h < H$. Because of the large entity vocabulary, having the entity embeddings be H dimensional would be highly memory intensive, hence why they are stored in a smaller latent space. A single, linear layer maps the entity embeddings from $\mathbb{R}^h$ to $\mathbb{R}^H$. The mapped entity embeddings are then added to the positional embeddings, making up the final entity embeddings. For the rest of the report, references to entity embeddings refer to this combined embedding, unless explicitly stated otherwise.

The produced word and entity embeddings are concatenated to a matrix $\in \mathbb{R}^{(n+m)\times H}$.

Transformer The embeddings of words and entities are forward passed through a transformer operating on entire sequences of embeddings both those representing words and entities. This component is dubbed the encoder and, after N attention blocks with the same input and output dimensionality, outputs the final contextualized H-dimensional representations: The first m correspond to words and the last n to entities. The main LUKE model, LUKE large, has, following BERT, $N = 24$ and $H = 1024$.

After the computation of the representations, the model can be extended with a decoder consisting of bi-directional classification heads for pretraining, or it can be extended with a linear layer for downstream tasks.

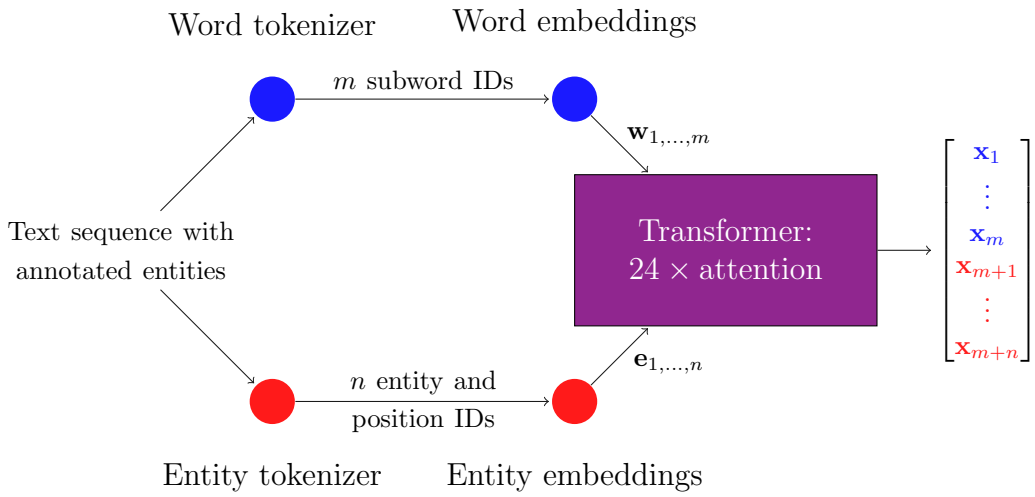

Figure 2.2: Overview of the LUKE pipeline from annotated text to contextualized representations of both words and entities. $\mathbf{w}_i, \mathbf{e}_i, \mathbf{x}_i$ all reside $\in \mathbb{R}^H$ with $H = 1024$ in the LUKE implementation.

Transfer Learning The word/entity duality of LUKE means that a large part of the model performs the same task as the conventional word-based LMs. For this reason, the word embeddings in LUKE follow the BERT architecture, and Yamada et al. initialize these embeddings to those found in RoBERTa. In the pretraining of Yamada et al., the encoder of LUKE is also equivalent to that of BERT and is initialized using the weights of RoBERTa. These transfer learned weights are kept fixed for the first half of the training.

Entity-aware Self-attention Additionally, Yamada et al. present an entity-related change to the BERT encoder architecture in the query mechanism of the attention scorer [Yam+20, Sec. 3.2].

For the attention between token i and token j with the respective representations $\mathbf{x}_i$ and $\mathbf{x}_j$, a core part of the normal transformer attention mechanism is to compute the following scalar:

$$q_{ij} = \mathbf{x}_j^\top \mathbf{Q} \mathbf{x}_i, \tag{2.1}$$

where $\mathbf{Q} \in \mathbb{R}^{H \times H}$ is a learnable component called the query matrix [Vas+17, Sec. 3.2.1].

In LUKE, the tokens $\mathbf{x}_i$ may either be words or entities. To handle this explicitly, the idea of *entity-aware self-attention* changes the computation of the query attention scalar q_{ij} to

$$q_{ij} = \begin{cases} \mathbf{x}_j^\top \mathbf{Q}_{w2w} \mathbf{x}_i & \text{if both } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are word tokens} \\ \mathbf{x}_j^\top \mathbf{Q}_{w2e} \mathbf{x}_i & \text{if } \mathbf{x}_i \text{ is word and } \mathbf{x}_j \text{ is entity} \\ \mathbf{x}_j^\top \mathbf{Q}_{e2w} \mathbf{x}_i & \text{if } \mathbf{x}_i \text{ is entity and } \mathbf{x}_j \text{ is word} \\ \mathbf{x}_j^\top \mathbf{Q}_{e2e} \mathbf{x}_i & \text{if both } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are entity tokens} \end{cases}. \tag{2.2}$$

LUKE, however, is not pretrained using this mechanism, but for the fine-tuning tasks, Yamada et al. show in ablation experiments that using this addition consistently yields better performance [Yam+20, Sec. 5.2]. This new mechanism, theorized to improve entity knowledge by Yamada et al., was not used in pretraining with the following explanation: "We perform pretraining using the original self-attention mechanism rather than our entity-aware self-attention mechanism because we want an ablation study of our mechanism but can not afford to run pretraining twice" [Yam+20, Sec. 3.4].

2.5.2 Pretraining

Yamada et. al perform the pretraining of LUKE by extending the MLM task of RoBERTa [Liu+19]. Where RoBERTa is trained to predict randomly masked words, LUKE is trained to predict both randomly masked words and entities, such as

- "Franklin Roosevelt did not [MASK] to see the [MASK] of WWII."
  Truth: "live", "end".
- {*[MASK]* at words #1-2, *World War II* at word #11}
  Truth: "Franklin D. Roosevelt".

Following RoBERTa [Liu+19], subword tokens making up 15 % of full words in each sequence are, each batch, randomly replaced with the [MASK] token. Of these, 10% are unmasked again, and another 10% are replaced by a random token from the vocabulary instead of [MASK]. Similarly, 15 % of entity tokens are replaced with [MASK], but none of these are unmasked or replaced with random entities. The model is then trained to predict tokens at mask positions. The model parameters are updated by the stochastic gradient optimization algorithm known as *Adam with weight decay fix* (AdamW) [LH19].

For a batch of size N and C classes, the model produces a matrix of size $N \times C$ where $X_{i,j}$ is the class score for class j for the i'th example in the batch. For each of the two classification tasks, the cross entropy loss l is calculated as

$$l = \frac{1}{N} \sum_{i=1}^{N} \left(-X_{i,c_i} + \log \sum_{j=1}^{C} \exp X_{i,j} \right), \tag{2.3}$$

where c_i is the true class of the i'th example [Conb]. The total loss is calculated as the sum of individual losses for both tasks.

Yamada et al. [Yam+20] use weight decay for the entire model, except for bias and layer normalizations. According to Krogh and Hertz [KH91], the regularization induced by weight decay improves generalizability in feed-forward networks.

Entity Mask Prediction Head For the entity pretraining task, LUKE is equipped with another classifier structure in addition to the MLM prediction head inherited from BERT. This new prediction head follows the architecture of the masked word scorer just operating on the entity representations. The masked entity tokens are thus scored as corresponding to one of the entities in the entity vocabulary by two linear layers.

The GeLU activation [HG20] and layer normalization components are placed between the layers.

2.5.3 Fine-tuning for Named Entity Recognition

Yamada et al. base their NER fine-tuning on the method from Sohrab and Miwa [SM18]. All possible spans over the entire sequence (or n-grams) are calculated and saved as entity candidates. Spans longer than 16 words and spans that cross sentence boundaries are discarded. Each span along with the text in which it appears is saved as a NER example.

An example is forward passed through LUKE by marking the candidate entity with the [MASK] entity ID. For each named entity candidate, the computed representation of this [MASK] entity and the CWRs of the first and last word tokens in the span are concatenated. [MASK] is used as NE candidate as this entity ID was used for a similar prediction task in the pretraining. For NER, only [MASK] and [PAD] are used out of all the entity embeddings that LUKE contains.

The combined representation from the pretrained model is then given to a single linear layer that learns to classify the candidate as either a non-entity or one of the classes in the dataset. For a dataset with n classes, the learning problem is an $n + 1$ class classification task. The loss is calculated as the cross entropy loss between the predicted class scores and the true class scores using equation (2.3). Both the full pretrained model and the linear classifier are fitted. An issue with the n-grams is the large amount of overlaps between spans. This is solved by greedily selecting spans based on the class scores from the classifier. Thus, if two (partially) overlapping spans both are predicted to cover an entity, only the one with the highest score is kept, while the other is predicted to be a non-entity.

3. Data

3.1 Entity-annotated Danish Wikipedia

For the pretraining of LUKE, a corpus of entity-annotated text is needed. Entity annotations consist of two things: The word span and the name of the entity. No text is written directly with such annotations in mind. However, some texts such as news articles often contain embedded hyperlinks that implicitly provide the word spans, but these often point to other articles rather than specific items. On Wikipedia, this is not the case, as hyperlinks point to articles on specific topics, which makes it suitable for LUKE's pretraining task. Given its size, Wikipedia is thus a particularly useful pretraining dataset.

As of June 23rd, the Danish Wikipedia has 267,408 content pages containing a total of 81,822,460 words[3.1]. The Danish Wikipedia from March 1st, 2021, which is marginally smaller, was used for building the DaLUKE dataset. Yamada et al. limit the entity vocabulary to the 500,000 most often linked pages [Yam+20]. As the Danish Wikipedia is only roughly 2.5% the size of the English Wikipedia, no limit on the entity vocabulary was imposed, to preserve data.

Every article is a sequence of words with embedded hyperlinks. These were split into sequences of at most 512 tokens using BotXO's Danish BERT tokenizer, and the token spans of the hyperlinks were saved. Following BERT [Dev+19], the first and last tokens were always [CLS] and [SEP], respectively, with the remaining 510 tokens coming directly from the tokenizer. Sequences were cut short if necessary to prevent hyperlinks spanning two sequences. All sequences contain text from at most one paragraph in an article.

Word spans over tokens were also saved, as the MLM pretraining task was done on a full word level. Subword and entity tokens were masked using the [MASK] token. The [UNK] special token is reserved for subwords that are not part of the tokenizer's vocabulary.

Thus, an entry in the dataset – called an example – consists of the following:

[3.1] `https://da.wikipedia.org/wiki/Speciel:Statistik`. Visited June 23, 2021.

- An array of at most 512 subwords token ID's
- An array of at most 128 entity ID's
- An array of two-tuples containing the start and end indices for every entity
- An array of two-tuples containing the start and end indices for every full word

Finally, the pretraining dataset was the set of all examples from all articles.

3.1.1 Entity Augmentations

One limitation of Wikipedia as a pretraining dataset is that hyperlinks are rarely repeated, even if the same article is referenced multiple times on a page. This results in false negatives – cases where entities are not annotated as such.

In order to augment the entity annotations, we performed preprocessing on Wikipedia before pretraining. Our preprocessing consisted of the following two steps applied article-wise:

1. Collect the set of mentioned articles including the title of the article itself. Only include titles of at least three characters.
2. For each title, check if it matches a sequence of characters following a whitespace while ignoring casing. If there is a match, annotate and go to next whitespace.

Many hyperlinks span only the first part of a given word. For instance, this allows the word "Windmills" to link to the article "Windmill". When matching, such cases were annotated as we expected it to remove numerous false negatives, while introducing relatively few false positives. Not annotating titles shorter than three characters prevented the article on "A" to be linked at every word starting with "A". We also only matched titles already linked in the articles, as we assumed that any article that should be linked to, had already been linked at least once. This should further reduce the number of false positives. Whether this had a positive effect on learning, is subject to an experiment presented in Section 6.1.5.

Table 3.1 shows how these augmentation steps change the properties of the dataset.

Statistic	Unaugmented	Augmented	Difference
Subword tokens	101,394,208	108,322,681	+6,8 %
Entity vocabulary size	214,150	225,198	+5,2 %
Entity annotations	4,883,918	7,180,372	+47,0 %
Examples	363,498	378,357	+4,1 %

Table 3.1: Statistics for both datasets. The increase in entity vocabulary size is explained by adding links that point to the article in which the link occurs. This means that some articles, which are never linked to, have links in the augmented dataset, causing to be included in the entity vocabulary. The increased number of subword tokens and examples comes from not including sentences without entities, of which there are fewer in the augmented dataset.

3.2 Named Entity Recognition Benchmarks

3.2.1 Danish Named Entity Recognition Datasets

For the task of benchmarking both existing and new Danish NER models, three publicly available datasets previously used in the literature were identified.

Danish Universal Dependencies Universal Dependencies of Danish Dependency Treebank (UD-DDT) is a grammatically annotated dataset produced by converting the Danish Dependency Treebank corpus [Kro03] to the Universal Dependency annotation format [JMP15]. The dataset consists of 5512 sentences and 100,733 characters from the Danish PAROLE corpus containing book, newspaper and journal literature from 1983-1992 [NKA98]. UD-DDT is split into training, development and test sets consisting of 4383, 564 and 565 sentences, respectively. Two NER annotations of this dataset are considered.

- *DaNE* is a 2020 NER annotation of the entire UD-DDT into four categories PER(son), ORG(anisation), LOC(ation), MISC(ellaneous) performed once by a linguist and once by a team of non-linguists for the Alexandra Institute [Hvi+20, Sec. 4]. The categories are described in closer detail in Section 3.2.3.
- A NER annotation of the development and test subsets of UD-DDT into the same four entity categories was performed by Barbara Plank in 2019 [Pla19]. A subset of the training data was also annotated. This annotation will be called *Plank*.

Wikipedia Annotations Pan et al. performed an automatic NER annotation of a Wikipedia corpus in 2017 by transferring English NER of the categories PER, ORG, and LOC to 281 languages including Danish [Pan+17]. This dataset is known as *WikiANN* (Wikipedia Annotations) and the Danish version includes 40,000 sentences. Pan et al. consider this a "silver-standard annotation" due to its automatic generation which they label knowledge base mining [Pan+17, p. 1946]. The balanced training, development, and test splits from Rahimi et al. are used [RLC19].

Comparison These three NER-annotated datasets were used and a summary of their counts is shown in Table 3.2. DaNE is considered the main public Danish NER dataset, as it is produced with the highest degree of expert supervision and used for training most of the models compared in Section 4.1.3. However, as shown on Table 3.3 and Figure 3.1, DaNE and Plank suffer from uneven label distributions in the different subsets. As most model training assumes that the training data has the same distribution as the true distribution, this could cause evaluations on the test sets to give a poor approximation of real-world performance. Previous literature has highlighted the scarcity of accessible, gold-standard, Danish NER data as a limitation in the field [Pla19, Sec. 2.1].

Dataset	Number of sentences			Test set entities			
	Train	Dev.	Test	LOC	PER	ORG	MISC
DaNE	4,383	564	565	96	180	161	121
Plank	604	564	565	97	169	94	30
WikiANN (da.)	$20 \cdot 10^3$	$10 \cdot 10^3$	$10 \cdot 10^3$	5,242	4,378	4,078	0

Table 3.2: Counts of sentences and test set entities in the three Danish NER benchmarks. Note the clear differences in annotations between DaNE and Plank, which use the same text corpus.

Subset of data	LOC	PER	ORG	MISC
Training	945	1,249	802	1,007
Dev.	111	166	90	113
Test	96	180	161	121

Table 3.3: The number of NEs by category and data split in DaNE.

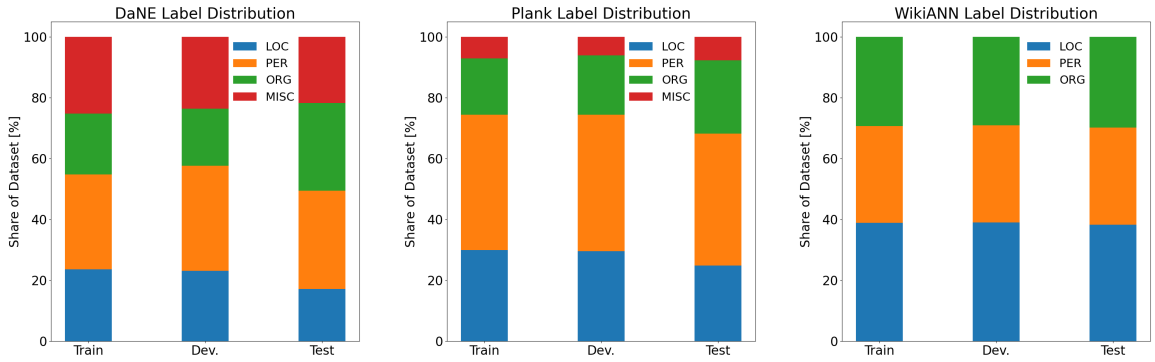

Figure 3.1: Distribution of entity labels in the three Danish NER datasets. Notably, DaNE and Plank do not have the same distributions in the three datasets, with e.g. LOC being under-represented in the DaNE test set and MISC being over-represented in the Plank test set. WikiANN is the only dataset that has the same distribution in all three subsets.

3.2.2 CoNLL-2003

The shared task for the 2003 Conference on Computational Natural Language Learning (CoNLL) was NER for which annotations on English Reuters news wire articles were performed by researchers [TD03]. This corpus consists of 1393 articles that were divided into training, development and test sets as seen in Table 3.4. The dataset is the central benchmark in English NER [Yam+20, Sec. 4.3] and has a competitive history with over 50 models submitted to the performance leaderboard at Papers With Code [3.2].

CoNLL-2003 English	Articles	Sentences	Tokens	LOC	PER	ORG	MISC
Training set	946	14,987	203,621	7,140	6,600	6,321	3,438
Development set	216	3,466	51,362	1,837	1,842	1,341	922
Test set	231	3,684	46,435	1,668	1,617	1,661	702

Table 3.4: Dataset counts for the canonical NER dataset CoNLL-2003. We note multiple differences in NE distributions compared to the Danish NER datasets shown at Table 3.2 and take this as an indication of the different corpus backgrounds.

[3.2] See `https://paperswithcode.com/sota/named-entity-recognition-ner-on-conll-2003`. As of June 2021, LUKE is second in the leaderboard.

3.2.3 Annotation Schemes: What Do The Tags Mean?

The DaNE and Plank sets refer to the CoNLL-2003 as their reference annotation scheme [Hvi+20, Sec. 4] [Pla19, Sec. 2.1]. WikiANN follows the same categories, excluding MISC. All datasets follow the Inside-outside-beginning (IOB) format that ascribes a word the "I-X" tag if it is inside a named entity of type X, "B-X" if it begins the entity, and "O" if it is outside named entities (e.g. the null class). This format proposed by Ramshaw and Marcus in 1995 is widely used but does allow entities to be nested or to overlap [RM95].

Text example	Antonio	Vivaldi	came	from	Venice
IOB annotation	B-PER	I-PER	O	O	B-LOC

The definition of the annotation categories are given in the CoNLL-2003 guidelines[3.3] and are summarized here:

- LOC: Geographical regions, natural locations, and public and commercial places. Also includes abstract places.
- PER: Names, aliases of people, animals, and fictional characters.
- ORG: Companies, brands, government bodies, movements, clubs, and subdivisions thereof.
- MISC: Titles of media, nationalities, languages, religions, ideologies, wars, slogans, eras, and adjectives and word combinations that are derived from one of the other tags.

To end the introduction of the data, some examples of NEs in each testing dataset are shown. Note that the all-important context of the words is not shown here.

- DaNE: helvede (LOC), Holland (LOC), Astrid Lindgren (PER), Odense Teater (ORG), det danske rigsfællesskab (ORG), afghansk (MISC), Camel (MISC)
- Plank: Bagdad (LOC), Bjarne (PER), USA (ORG), Københavns Kommune (ORG), DR-dokumentar (MISC), Levi's jeans (MISC)
- WikiANN (da.): Dinariske Alper (LOC), Esbjerg Kommune (LOC), Jorge Luis Borges (PER), FC Barcelona (ORG), kommunehospitals (ORG)
- CoNLL-2003: Buenos Aires (LOC), Whistler Mountain (LOC), Katja Seizinger (PER), FIFA (ORG), Asian Cup (MISC), Swede (MISC)

[3.3] These are available at `clips.uantwerpen.be/conll2003/ner/annotation.txt`. Visited February 27, 2021.

4. Methods

4.1 Benchmarking Named Entity Recognition

4.1.1 Evaluation of Named Entity Recognition

As explained in Section 3.2.3, all datasets used in this project supply an annotation label in IOB format [RM95] for each word in each sequence. Comparing a sequence of true annotations and predictions, there are a number of potential error patterns as both the span and the type of the entity are to be guessed, see Table 4.1.

Sentence	Gaul	is	divided	into	three	parts.
True annotation	B-LOC	O	O	O	O	O
I (TP): Complete match	B-LOC	O	O	O	O	O
II (FP): Spurious entity	B-LOC	O	O	O	O	B-MISC
III (FN): Missing entity	O	O	O	O	O	O
IV: Wrong type	B-ORG	O	O	O	O	O
V: Wrong boundary	B-LOC	I-LOC	O	O	O	O
VI: Wrong type and boundary	B-ORG	I-ORG	O	O	O	O

Table 4.1: The different outcomes of a prediction when related to the ground truth. This categorization and naming is taken from [Bat18]. The first three simple cases can be translated to true positives, false positives, and false negatives, when considering only one entity type at a time.

In our evaluation, the simple and strict approach of the CoNLL-2003 shared task evaluation was followed, explained as:

"Precision is the percentage of named entities found by the learning system that are correct. Recall is the percentage of named entities present in the corpus that are found by the system. A named entity is correct only if it is an exact match of the corresponding entity in the data file." [TD03, Sec 2.4]

Using the terminology of Table 4.1, these measures are below defined for a class (such as LOC) when assuming the predictions and ground truth both contain at least

one positive label.

$$\text{Precision} = \frac{\#\text{ case I}}{\#\text{ case I} + \#\text{ case II}}, \quad \text{Recall} = \frac{\#\text{ case I}}{\#\text{ case I} + \#\text{ case III}}. \tag{4.1}$$

From these class-level metrics, global values of precision and recall were reported using the *micro average*, that is, by aggregating the number of prediction outcomes (I, II, III) across classes before using eq. (4.1). In the report, every reference to an average of metrics over NER classes refers to this micro average. Thus, using this approach, the error types IV, V and VI are not treated explicitly.[4.1].

Following previous literature, the main score on which the system performance was judged is the F1 score, the harmonic mean of precision of recall;

$$\text{F1} = \frac{2}{\text{Precision}^{-1} + \text{Recall}^{-1}} \tag{4.2}$$

The reported scores were obtained using a Python sequence evaluation library [Nak18], also used by the original LUKE paper and DaNLP, with whom comparisons are made.

Two of the three used datasets contains MISC annotations, and existing literature disagrees on whether to consider this class in evaluations, with DaNLP arguing "We are only reporting the scores on the LOC, ORG, and PER entities as the MISC category has limited practical use" [Bro+21, Sec. "Benchmarks" in NER page]. We, however, trained all models with the MISC annotation and also included it in our evaluations, as we focus less on the practical application of the measure and more on its connection to language understanding. When comparing with models trained without MISC, a non-MISC average F1 score is also reported (by treating all MISC predictions as null class predictions).

When comparisons are made between model average F1 scores, an approximate 95% confidence interval is reported (every time $\pm$ was used). This is calculated using the normality approximation for binomial observations [Bro+18, Method 7.3], which gives,

[4.1] Using this approach, the cases IV and VI in Table 4.1 would for the evaluation of the LOC class be treated as case III and for the evaluation of the ORG class be treated as case II. V would, similarly, be counted as both a case II and III for ORG. Such errors would thus be judged harshly in the global metrics and not show up in any way as partial corrects for any of the classes.

when denoting the true avg. F1 score f and the n sample estimate thereof $\hat{f}$, that

$$I = \hat{f} \pm z_{0.975} \sqrt{\frac{\hat{f}\left(1-\hat{f}\right)}{n}} \Rightarrow \mathbb{P}\left(f \in I\right) \approx 95\,\%, \tag{4.3}$$

where $z_{0.975}$ is the 97.5 % standard normal distribution percentile. For the sample size n, the number of true entities[4.2] in the dataset is used – this is 558 for DaNE, 390 for Plank and 13698 for WikiANN.

4.1.2 Fine-tuning English LUKE

Yamada et al. benchmark LUKE on named entity recognition using the CoNLL-2003 dataset [Yam+20]. Reproduction of these results was attempted by obtaining the pretrained LUKE models from Yamada et al.'s software repository. The same hyperparameters as Yamada et al. were used and are shown along with the technical details in Table 4.2.

The fine-tuning procedure was repeated five times for each of the two released LUKE models, called *large* and *base*, to examine variability in the downstream training.

Pretrained model	LUKE large[4.3]	LUKE base[4.3]
Pretrained model parameters	$483 \cdot 10^6$	$253 \cdot 10^6$
Pretrained model entity vocabulary	$500 \cdot 10^3$	$500 \cdot 10^3$
Learning rate	10^{-5}	$5 \cdot 10^{-5}$
Effective batch size	16	16
Numeric precision	Mixed FP16/FP32 (Nvidia APEX)	
Training code	PyTorch-based `luke`-repository[4.4]	
Software version	Python 3.6, PyTorch 1.2	

Table 4.2: Hyperparameters used to fine-tune LUKE large and LUKE base on the CoNLL-2003 dataset.

[4.2]Here, the number of true, entire entity spans in each class is used, not the number of words annotated with the class; a unit used for other analysis in the discussion. The latter would overstate the number of independent observations as two words in the same entity share much information.

[4.3] The pretrained models were downloaded on 17/02-2021 from the LUKE software repository: `https://github.com/studio-ousia/luke/tree/6feefe657d97d2f847ace87f61f23b705f75d2aa#released-models`

[4.4]The repository `github.com/studio-ousia/luke` was cloned at commit-SHA `6feefe6`, installed, and used for the fine-tuning.

4.1.3 Off-the-shelf, Danish models

Nine publicly available Danish NER models usable by NLP practitioners were collected and evaluated on the testing datasets of the three Danish NER annotations, considered in Section 3.2.1: DaNE, Plank and WikiANN.

Most of the models were found through DaNLP [Bro+21], and a number of them were introduced in Section 2.3.

- **DaNLP da-BERT**: The da-BERT model [Bot19] fine-tuned for NER on DaNE by DaNLP.
- **NERDA m-BERT**: The multilingual BERT base released by the Google Research BERT team [Dev+19] fine-tuned by NERDA [KN20] on DaNE.
- **NERDA Ælæctra**: The transformer Ælæctra, released by Malte Højmark-Bertelsen [Høj20], fine-tuned by NERDA on DaNE.
- **DaCy**: An adaption of version 3 of the SpaCy framework [Hon+20] to Danish by Kenneth Enevoldsen including fine-tuning of on DaNE [Ene21]. Both the medium and large versions are benchmarked, using the base and large multilingual RoBERTa [Con+20] models, respectively. This 2021 model, large version, is the currently reported state of the art [Ene21; Bro+21].
- **DaNLP spaCy**: SpaCy, version 2, adapted to Danish by DaNLP and fine-tuned on DaNE.
- **DaNLP Flair**: The Flair framework [Akb+19] also adapted to Danish by DaNLP and fine-tuned on DaNE.
- **Polyglot**: An NLP framework supporting a wide range of tasks in many languages including NER in 40 different languages. The NER model is fine-tuned using automatic, language-agnostic annotations generated from Wikipedia and Freebase link structures [Al-+15].
- **daner (DKIE Stanford CRF)**: An application of the Stanford CoreNLP Conditional Random Field (CRF) NER classifier [Man+14] released by Leon Derczynski as a part of the DKIE project [DFB14]. The model was trained on NER

annotations produced at ITU on the Danish Dependency Treebank (DDT) corpus [Kro03]. The released Java-based NER tool is called `daner`[4.5].

The weights of all these finished NER models were downloaded and evaluated following Section 4.1.1. The code for performing inference for all the models on the three datasets and measuring the F1 scores can be found in the module `reproduction.danish_ner` in the DaLUKE repository[4.6]. The reproduction was performed using Python version 3.7.10 and the dependency versions defined in the requirements of DaNLP and NERDA.

4.2 DaLUKE

4.2.1 Pretraining Methodology and Hyperparameters

Our pretraining approach followed the one used by Yamada et al., as described in Section 2.5.2, with four key differences:

1. Entity-aware self-attention was used in the pretraining, as this is expected to positively impact the modeling of knowledge. This speculation is tested in Section 6.1.4.
2. The entity-augmented Danish Wikipedia described in section 3.1.1 was used.
3. The model followed the architecture of BERT base instead of BERT large, and weights were initialized from BotXO's da-BERT [Bot19] as no Danish RoBERTa or BERT large is available. An ablation study on this transfer learning from da-BERT is introduced in Section 6.1.6. Because of this, token embeddings and contextualized representations both reside in a 768-dimensional latent space rather than the 1024 dimensions used by LUKE large. DaLUKE also has 12 attention blocks rather than 24. [Yam+20, Sec. 3.4]
4. The entity vocabulary, consisting of all mentioned Wikipedia articles, was limited in the English LUKE for computational efficiency [Yam+20, Sec. 3.4]. We do not perform such limiting as the number of mentioned Wikipedia articles is much smaller in the Danish corpus. An experiment of limiting the vocabulary is discussed in Section 6.1.7.

[4.5] The repository is at `github.com/ITUnlp/daner`

[4.6] The reproduction code is available here: `github.com/peleiden/daluke/tree/master/reproduction/danish_ner`.

Due to the memory-intensive nature of the transformer architecture, we followed Yamada et al. and used gradient accumulation over multiple subbatches within each batch, meaning that each parameter update consists of multiple forward passes of the model.

As with LUKE, the cross entropy loss is calculated using eq. (2.3) for each classification task, with the only difference being that the final loss is the average rather than the sum of the individual task losses. This effectively scales the loss and does not affect the learning if the learning rate is adjusted accordingly.

The hyperparameters used for pretraining are shown in Table 4.3. These were chosen based on informal experiments, as structured hyperparameter search was not possible within the time frame of this project.

Parameter	Value
Epochs	150
Batch size	4080
Peak learning rate	$3 \cdot 10^{-4}$
LR warmup steps prop.	6 %
Mask prob. for words	15 %
Mask prob. for entities	15 %
Dropout	0.1
Weight decay	0.01
AdamW β_1	0.9
AdamW β_2	0.999
AdamW ϵ	10^{-6}

Table 4.3: Hyperparameters used for DaLUKE pretraining.

LUKE follows BERT and increases its learning rate linearly from 0 to the peak learning rate followed by a linear decrease to 0 for the rest of the training - the slanted triangle learning rate (STLR). [Dev+19; Yam+20; HR18]

For pretraining, DaLUKE also employs linear warmup for the first 6 % of parameter updates, after which it decreases polynomially with a power of $\sqrt{3}$ to a final learning rate at 10 % of peak learning rate. This results in a more aggressive decrease in learning rate after the warmup period but a slightly higher learning rate in the final steps compared to STLR. This was used as the amount of available compute was not known in advance, and this approach would work well with both early stopping and continued training. The learning rate development is shown on Figure 4.1.

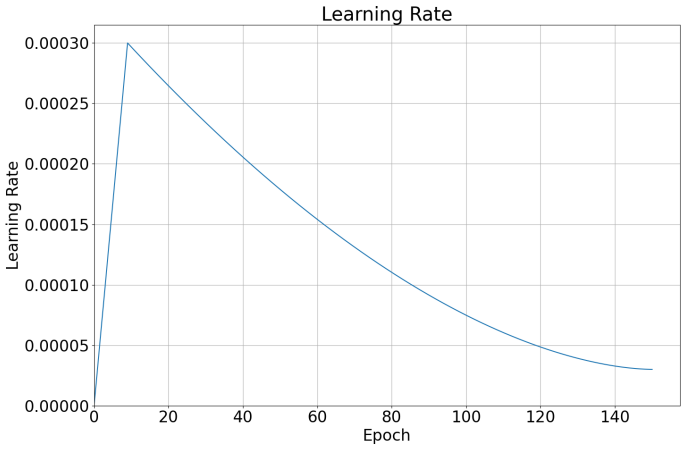

Figure 4.1: Learning rate during the pretraining.

Following Yamada et al., we do not update the parameters of the weights transferred from da-BERT for the first half of the training.

Accuracy measures To measure the performance of the pretraining model, the accuracies of word and entity predictions were calculated throughout the training. Due to the size of the classification problem, especially regarding entities, top k accuracies were calculated - that is, whether the true token was among the k tokens given the highest probability by the model.

4.2.2 Fine-tuning DaLUKE for Named Entity Recognition

The fine-tuning of DaLUKE largely followed that of LUKE, described in Section 2.5. The main DaLUKE NER model was produced by fine-tuning on DaNE.

After each epoch, the model was evaluated on the development split of the dataset. When a better score than the so far best checkpoint was found, this was overwritten. The best checkpoint of the model was used for evaluation on the test set.

Learning rate Unlike in the pretraining, we did as Yamada et al. and used a STLR [Yam+20]. Howard and Ruder have also shown that the STLR approach for fine-tuning tasks improves results compared to a linear decaying learning rates without warmup [HR18].

Loss Due to the nature of the n-grams, only very few spans correspond to entities, making the O class dominate the dataset. For instance, in the training set of DaNE, the O class accounts for 99.3 % of spans, and the entity counts are not balanced. This motivated us to implement class frequency-weighted loss, the performance of which is discussed in Section 6.2.2.

The class-weighted loss l_w was calculated similarly to the unweighted loss in eq. (2.3):

$$l_w = \frac{1}{\sum_{j=1}^{C} w_j} \sum_{i=1}^{N} w_{c_i} \left(-X_{i,c_i} + \log \sum_{j=1}^{C} \exp X_{i,j} \right) \tag{4.4}$$

where w_j is the weight of class j [Conb]. We used the reciprocal of the number of occurrences of each class in the training dataset.

Unless otherwise stated, all fine-tunings presented in the rest of the report use the non-weighted loss.

Hyperparameter search The development set of DaNE was used to guide a simple search for fine-tuning hyperparameters for the final model. The search is performed by repeating the training procedure with all combinations of the following parameter values that were chosen based on previous, informal experimentation.

1. Batch size $\in \{8, 16\}$
2. Peak learning rate $\in \{10^{-5}, 5 \cdot 10^{-5}\}$
3. Dropout (final, linear layer) $\in \{0.025, 0.1\}$
4. Whether to use class-weighted loss $\in \{\text{Yes}, \text{No}\}$

Parameter	Value
Epochs	15
Batch size	8
Peak learning rate	$5 \cdot 10^{-5}$
LR warmup steps proportion	6 %
Dropout (pretrained model)	0.1
Dropout (final, linear layer)	0.025
Weight decay	0.01
Loss weighting	No

Table 4.4: Hyperparameters used for fine-tuning DaLUKE for NER.

For each combination, the resulting model was evaluated on the development set, shown in Table A.3, and the hyperparameters with the highest average F1 score were chosen. The final model was produced by fine-tuning again to limit selection bias. The chosen hyperparameters are reported in Table 4.4 together with the used hyperparameters that were not tested.

4.2.3 Implementation Details and Open Source Software

Open Source Software Package We publish DaLUKE and all related code under the MIT open source software license [MIT]. All code and documentation is available at `https://github.com/peleiden/daluke`. The pretrained model is available with MLM layers at `https://nx5746.your-storageshare.de/s/qDM2TE6m9CKmPWd`. The model fine-tuned on DaNE is available at `https://nx5746.your-storageshare.de/s/wxbY3TrwfAoYxb9`. Our software package is `pip` installable and runs on Python 3.8 and above. Simply run `pip install daluke` to install it locally. It allows the use of DaLUKE both for MLM and NER on text files or inputted strings directly from the command line. For instructions, see the readme file on the repository.

Implementation Yamada et al. have made their code[4.7] freely available under the Apache License 2.0 [Fou04]. We highly appreciate this, and our original plan was to use their code. However, we quickly ran into problems with this approach, as getting all dependencies installed with the correct versions on our computing cluster proved to be a very challenging task. To solve this, we forked the repository and updated both Python and several of the dependencies to newer versions, but this in itself required significant code rewrites. When that worked, we started pretraining, which presented more mysterious issues: Training on a single GPU worked, but two GPUs slowed training by a factor of three, and three or more GPUs simply did not work. Due to limited GPU availability, cryptic error messages, and a lack of documentation, debugging once again became difficult.

These reasons compelled us to reimplement the model and training from scratch, which also allowed fitting the software to our specific needs. The forked LUKE repository, available at `https://github.com/peleiden/luke`, is used for parts of the pretraining dataset preparation, but everything else has been reimplemented. Our code

[4.7] `https://github.com/studio-ousia/luke`. Visited June 11, 2021.

is inspired by the original LUKE code, but we have made efforts to make it more approachable focusing on code readability and documentation.

Rewriting code bases always carries a risk of introducing bugs. Despite borrowing heavily from Yamada et al., we did introduce some errors in the code that had to be fixed, forcing multiple restarts of the training. These ranged from domain-specific ones such as accidentally leaving category pages in the entity vocabulary to general errors such as copying pointers instead of the memory stored at them. Our answer to iron these out was unit testing of modular code, but we still have work to do in expanding test coverage.

Floating Point Precision The model was trained with PyTorch Automatic Mixed Precision (AMP) [Cona], which – ideally, at least – should decrease training time with little to no penalty to accuracy [HTC20]. This is partially due to half-precision calculations being simpler, and partially due to the lower memory requirements, allowing larger subbatches and thus better GPU utilization. However, as Figure 4.2 shows, AMP works as intended when training on NVIDIA Tesla V100s, but had the opposite effect when training on NVIDIA A100s.

AMP works by automatically casting parts of the model to half-precision rather than the usual single precision. Some layers are more sensitive than others to precision. For instance, linear layers are always cast to half-precision, but the loss is calculated using single precision. Due to the smaller precision, underflow in the gradients is a risk. This is handled by scaling up the loss and then scaling down the gradients accordingly. [Cona]

Distributed Training and Runtime As transformer training requires significant compute, distributing the training over multiple GPUs can yield considerable speedups. Especially given the parallelizable nature of the transformer compared to recurrent neural network variations, training should scale fairly well on multiple GPUs [Vas+17].

The pretraining took roughly a week. Due to varying GPU availability, varying numbers of A100s and V100s were used for pretraining and the experiments in section 6.1. The main pretraining took approximately a week and used 2-4 V100s. For comparison, Yamada et al. trained LUKE using 16 V100s for 30 days. [Yam+20]

We measured the time needed to train for one epoch on different cluster configurations. The results are shown on Figure 4.2.

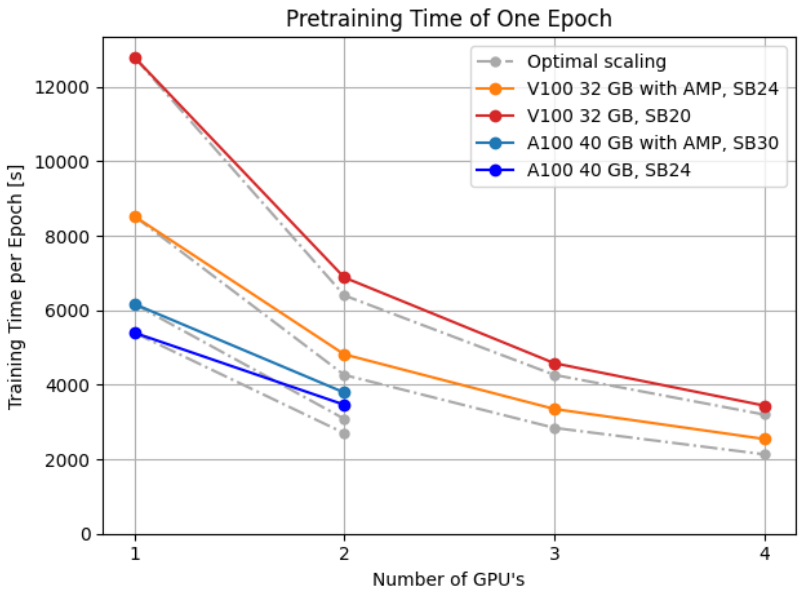

Figure 4.2: How GPU models, the number of GPUs, and AMP influences pretraining runtime. Sub-batch size (SB) is included, as it is important for GPU utilization. The measurements were taken with da-BERT weights locked. The V100s act mostly as expected with close to optimal scaling and AMP decreasing runtime. Surprisingly, however, the A100s, while faster, scale poorly and are slower when using AMP. The poor scaling is partially explained by Amdahl's law [Kle11]: The sequential parts of the code do not get faster along with the GPU and so take up a relatively larger amount of the runtime. The A100s also do not have an NVLink bridge unlike the V100s and must therefore communicate over PCIe, which is slower. The poor AMP performance, however, is harder to explain. According to Hans Henrik Sørensen, DTU Computing Center, it is caused by different `gemm` implementations, but we have not investigated it any further. Python 3.8.4 using PyTorch 1.8.1 compiled with CUDA 11.1 was used.

5. Results

5.1 English LUKE Reproduction

Model	Over 5 repetitions	Micro avg.	LOC	PER	ORG	MISC
LUKE large	Mean F1 [%]	94.0	95.0	97.2	93.6	85.2
	Std. [%]	0.2	0.1	0.1	0.3	1.0
LUKE base	Mean F1 [%]	93.4	94.6	96.8	92.4	84.9
	Std. [%]	0.2	0.2	0.2	0.2	0.7

Table 5.1: Observed English LUKE results over five repetitions of fine-tuning and evaluating LUKE on CoNLL-2003 for each model size.

Yamada et al. report micro avg. F1 scores of 94.3 % and 93.3 % for LUKE large and base, respectively [Yam+20]. In both cases, the reported scores are within two standard deviations on observed F1 scores, which were estimated on training repetition. This is expectable, and we conclude that the reproduction was successful. However, the variability of fine-tuning is also highlighted here and will be discussed in Section 6.2.

5.2 Pretraining of DaLUKE

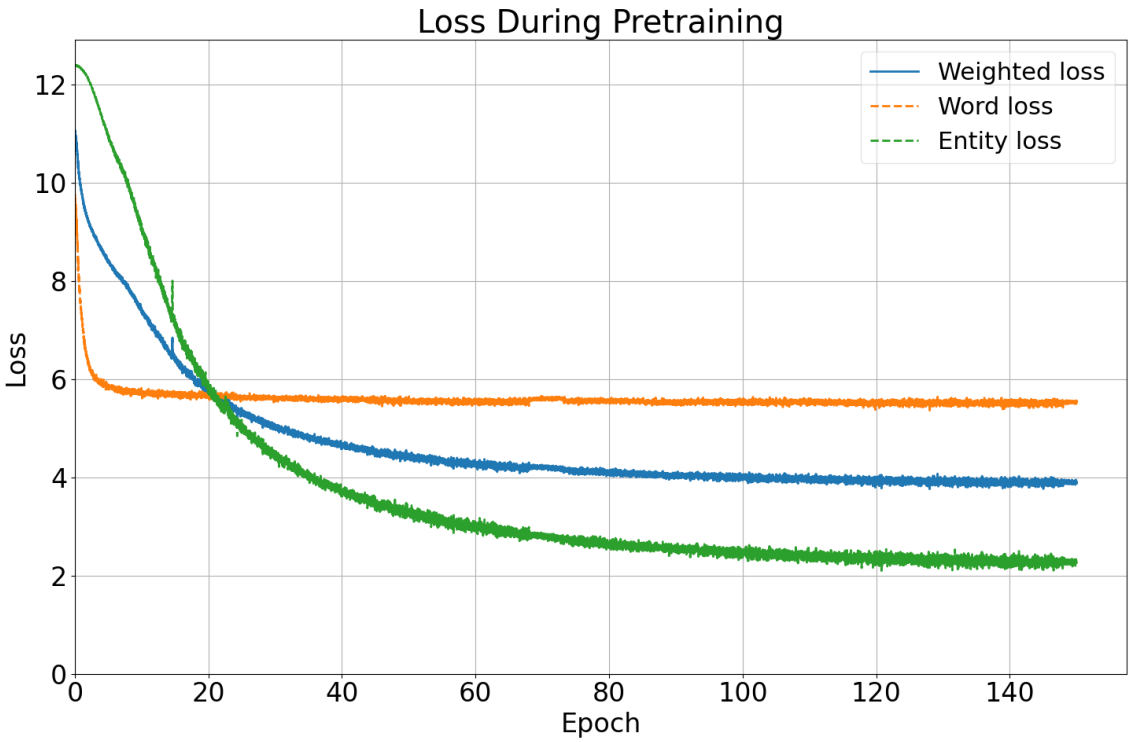

Figure 5.1: Cross entropy loss throughout pretraining for both masking tasks as well as effective (weighted) loss, which is the average of the two. The loss on the entity masking (green) seems to drive close to all development in the average loss (blue) after the first few epochs.

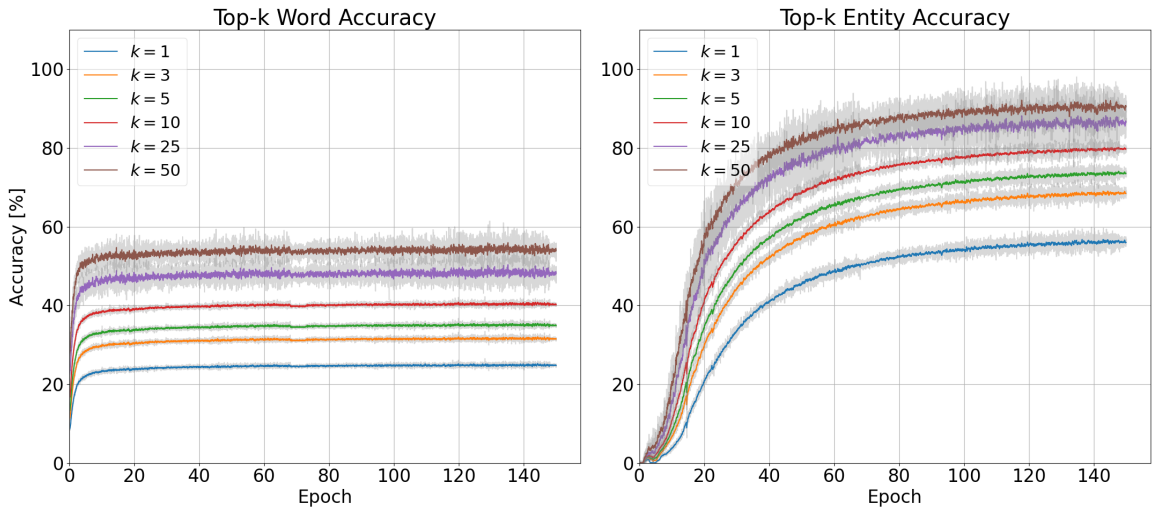

Figure 5.2: Masked words and masked entity accuracy throughout pretraining. The curves are smoothed with a rolling average. The frequency of exact correct word guesses quickly rests at 20-25 % while the new task of entity masking starts from 0 and takes time to learn.

The loss shown on Figure 5.1 mirrors the accuracy with masked word loss converging quickly, after which falling entity loss becomes the driving factor behind the falling total loss. The masked words and masked entity accuracies are shown on Figure 5.2. The former converges quickly, achieving close to top accuracy in less than a tenth of the full training time, while the latter keeps improving throughout the training. That the MLM performance from BERT is quickly maximized is, however, perhaps not surprising, as many of the relevant weights were initialized from an already trained model which has been pretrained on the same task.

Top k accuracy [%]	$k = 1$	$k = 3$	$k = 5$	$k = 10$	$k = 25$	$k = 50$
Masked words	24.8	31.5	35.0	40.3	48.2	54.0
Masked entities	56.2	68.6	73.6	79.8	86.4	90.3

Table 5.2: The main pretrained model performance in the last of 150 epochs. We note that the model has learned to guess masked entities consistently as the correct entity very often is one of the top 50 guesses.

5.3 Danish Named Entity Recognition

5.3.1 Main Benchmark: Our Results and Reproduction

Model name	Trained on	Micro Avg. [%]				Class F1 [%]			
		F1	Prec.	Rec.	F1 $_{\text{MISC}}$	LOC	PER	ORG	MISC
DaLUKE	DaNE	82.9 ±3	84.7	81.2	85.2 ±3	**87.0**	**94.2**	73.2	74.6
DaNLP da-BERT	DaNE	–	–	–	84.0 ±3	83.9	92.8	73.0	–
NERDA m-BERT	DaNE	79.2 ±3	82.1	76.5	81.7 ±4	83.5	92.6	66.9	70.3
NERDA Ælæctra	DaNE	70.6 ±4	76.1	65.8	74.5 ±4	77.3	86.9	56.2	56.4
DaCy medium	DaNE	78.3 ±3	78.3	78.3	80.5 ±4	84.0	90.4	66.2	70.1
DaCy large	DaNE	**84.9** ±3	**86.2**	**83.7**	**86.9**±3	85.3	**94.2**	**79.0**	**78.1**
DaNLP spaCy	DaNE	73.8 ±4	76.1	71.5	75.7 ±4	76.0	87.8	59.6	66.1
DaNLP Flair	DaNE	–	–	–	81.8 ±4	84.8	93.2	63.0	–
Polyglot	Wikipedia	–	–	–	64.2 ±4	65.0	78.7	39.3	–
daner	ITU DDT	–	–	–	56.5 ±5	59.4	70.4	28.3	–

Table 5.3: Results of Danish NER models of the DaNE [Hvi+20] testing dataset consisting of 565 sentences. ± on the average F1's is notation for the approximate 95 % confidence interval calculated using eq. (4.3). Missing numbers are due to some models not being trained on the MISC annotations. We see the transformer-based models dominate the scores, spearheaded by the large implementation of DaCy.

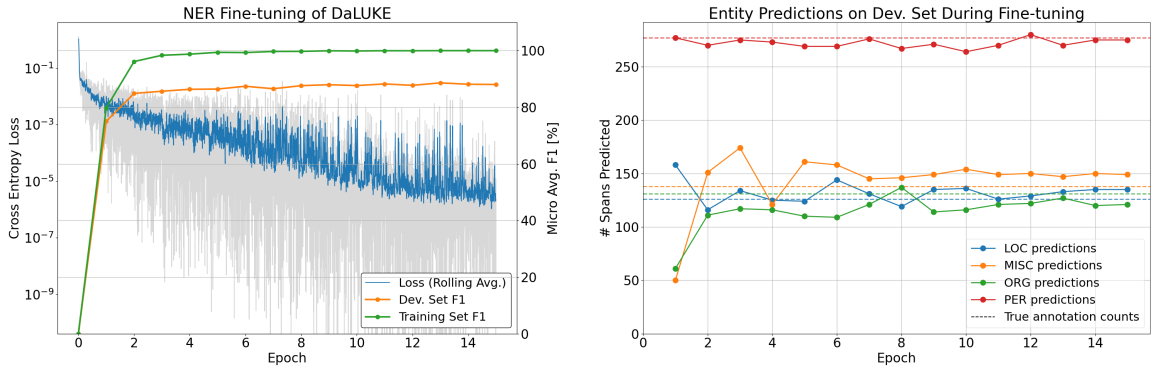

Figure 5.3: To the left, the development of loss and accuracy on the DaNE training and dev. sets. Almost all learning happens in the first few epochs, hinting that the pretrained model already holds much of the language knowledge used for NER. To the right, the number of predictions of each class on the dev. set is shown along with the actual number of each class where MISC and ORG initially are under-predicted.

We conclude, when comparing with the June 2021 DaNLP NER table[5.1][Bro+21], that the results of all NER results previously reported on DaNE are successfully reproduced with most scores corresponding exactly and a few varying within a percentage point.

Taking these results at face value, DaLUKE fails to achieve SOTA on DaNE, being beaten only by DaCy large but surpassing da-BERT, on which it is based. However, when considering the approximate uncertainty estimates, the small test set makes these conclusions baseless, instead identifying a *group* of transformer-based NER models leading the scoreboard.

Still, the considerable improvements in some categories when comparing to da-BERT is taken as an encouragement, suggesting that the extra pretraining and entity understanding indeed has added some knowledge to the model. The improvements of DaLUKE compared to da-BERT are examined further in the following chapter. Additionally, it is noted from considering the differences between algorithm performance across classes that future analysis of DaLUKE should examine how entity class performance is related to article bias in the Wikipedia pretraining dataset.

Ultimately, DaLUKE performs at a high level and shows promise for entity modeling.

5.3.2 Additional Datasets

Model name	Trained on	Micro Avg. [%]				Class F1 [%]			
		F1	Prec.	Rec.	F1 $_{\text{MISC}}$	LOC	PER	ORG	MISC
DaLUKE	DaNE	**69.0** ±5	**60.5**	80.3	78.9 ±4	**81.7**	**92.5**	56.0	13.2
DaNLP da-BERT	DaNE	–	–	–	**79.2**±4	78.6	93.4	**56.9**	–
NERDA m-BERT	DaNE	66.4 ±5	58.1	77.4	76.6 ±4	76.3	92.1	52.5	12.4
NERDA Ælæctra	DaNE	66.3 ±5	60.0	74.1	76.1 ±4	74.9	90.3	53.0	13.2
DaCy medium	DaNE	65.6 ±5	55.7	79.7	75.0 ±4	76.1	92.1	48.7	12.6
DaCy large	DaNE	68.5 ±5	58.9	**81.8**	79.0 ±4	79.0	**92.5**	58.0	**15.5**
DaNLP spaCy	DaNE	64.1 ±5	55.9	75.1	72.7 ±5	72.7	88.3	46.5	12.3
DaNLP Flair	DaNE	–	–	–	76.2 ±4	80.2	94.4	36.1	–
Polyglot	Wikipedia	–	–	–	64.1 ±5	69.7	78.4	24.7	–
daner	ITU DDT	–	–	–	59.8 ±5	58.2	73.6	26.1	–

Table 5.4: Danish NER algorithm performance on the Plank [Pla19] test set consisting of 565 sentences. These models fine-tuned on DaNE perform very poorly on the Plank MISC entities in particular, highlighting NE annotater disagreement.

[5.1]Table is available at `github.com/alexandrainst/danlp/blob/master/docs/docs/tasks/ner.md` and was compared on 27/6, 2021 at commit SHA `584d5c7`.

Model name	Trained on	Micro Avg. [%]			Class F1 [%]		
		F1	Prec.	Rec.	LOC	PER	ORG
DaLUKE	DaNE	**66.8**±0.1	**72.7**	61.9	73.5	74.9	44.2
DaNLP da-BERT	DaNE	65.7 ±0.1	68.6	63.2	72.1	74.5	40.1
NERDA m-BERT	DaNE	63.4 ±0.1	61.3	65.7	70.7	76.9	48.4
NERDA Ælæctra	DaNE	48.7 ±0.1	48.3	49.0	56.6	69.9	24.0
DaCy medium	DaNE	60.1 ±0.1	58.4	61.8	70.7	73.7	39.3
DaCy large	DaNE	64.6 ±0.1	62.2	**67.1**	**74.1**	**77.2**	**49.8**
DaNLP spaCy	DaNE	59.6 ±0.1	58.5	60.6	68.7	71.6	38.8
DaNLP Flair	DaNE	65.0 ±0.1	70.7	60.2	70.1	74.4	43.7
Polyglot	Wikipedia	62.9 ±0.1	66.3	58.2	72.4	69.2	35.3
daner	ITU DDT	46.6 ±0.1	51.5	42.5	56.2	54.5	14.8

Table 5.5: Results of Danish NER models on the WikiANN [Pan+17; RLC19] test set consisting of 10,000 sentences. This large dataset containing $\sim 14 \cdot 10^3$ entities results in small finite sample uncertainties on model performance making DaLUKE appear as a clear winner on this Wikipedia-based dataset.

When fine-tuned on DaNE, DaLUKE achieves SOTA on WikiANN and slightly tops DaCy large on Plank. This is unexpected given that DaLUKE was beaten on the DaNE test set. It may indicate a better generalization ability, which, if true, is indeed a desirable property. However, less exciting reasons for this behaviour may well be more likely. With Plank, the difference is minuscule and clearly within random error. The performance discrepancy on WikiANN appears surely significant, but may have an even less interesting reason: DaLUKE is pretrained directly on similar, annotated Wikipedia data, compromising the principle that the training and test sets should be strictly disjoint.

6. Experiments and Discussion

6.1 What Is Going on in the Pretraining?

To investigate what affects effectiveness of the pretraining, several ablation studies are performed. Unless stated otherwise, the main hyperparameters from Table 4.3 are used, but with 50 epochs due to limited compute. As the experiments were trained on a 1×A100 configuration, which exhibits issues with AMP, single precision was used for them all.

6.1.1 The Parameter Population

The first model analysis is a global view of the DaLUKE parameters. Figure 6.1 shows the model parameter value distribution before and after pretraining divided by whether a given parameter was initialized from da-BERT.

Two observations stand out: Firstly, DaLUKE-exclusive parameters, which consist of entity embeddings and the entity part of the decoder, are spread out. This includes the notable spike at the left figure that comes from biases and layer normalizations being initialized to 0. Secondly, the shared parameters are more or less unchanged in their distribution. This makes sense, as they are already trained, hence further training on similar data with a similar task should not result in any significant change. Furthermore, all parameters from da-BERT (barring the three sets of DaLUKE-exclusive query matrices) are not changed for the first half of the training. When they finally are unlocked, both the loss and learning rate have dropped significantly, resulting in less change.

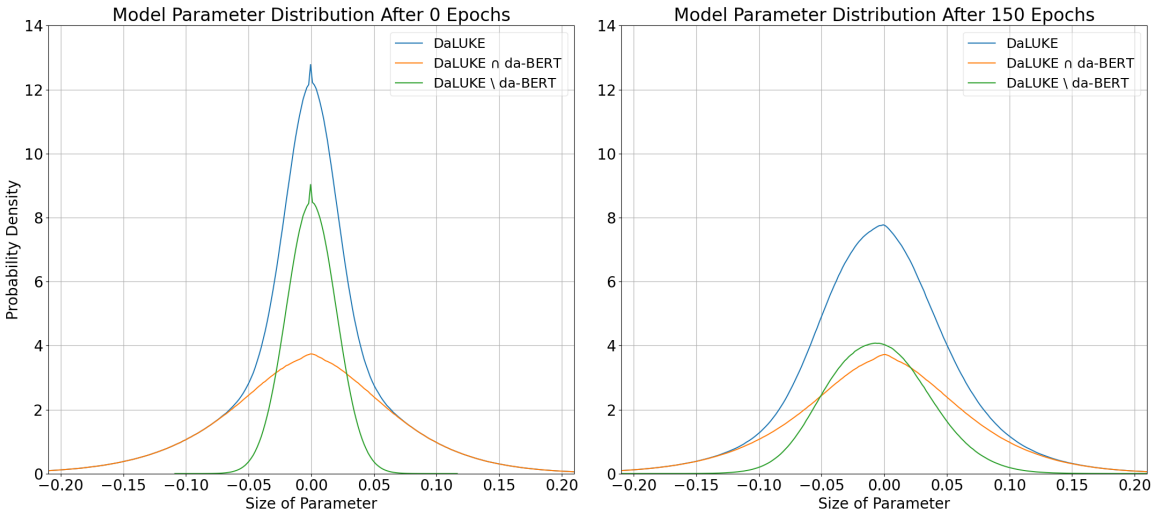

Figure 6.1: The parameter distribution before and after pretraining, including the decoder. DaLUKE ∩ da-BERT (orange) refers to the parameters that transferred from da-BERT, including the query matrices in the entity-aware self-attention. DaLUKE \ da-BERT (green) are the remaining parameters. These are made up of the entity embeddings (58 million parameters) and the entity decoder (58 million parameters). In total, the model has 273 million parameters. Note that DaLUKE ∩ da-BERT and DaLUKE \ da-BERT are not full probability densities but rather subsets that together make up the DaLUKE graph, which is a full probability density.

It is noted that the new DaLUKE parameters after training form a distribution appearing Gaussian. These are mostly made up of entity embeddings subjected to layer normalizations which we suspect to be influential on this shape.

6.1.2 Effect of More Pretraining

The goal of the pretraining is to infuse the model with general language understanding, but it is not immediately clear how to measure such an abstract concept. The simplest idea is to use the loss and, by extension, the top k accuracies, but this approach is not exempt of issues: Without a validation set, overfitting can be disguised as improvement, especially on small datasets. Furthermore, decreases in loss could from the decoder part of the MLM, which is irrelevant to downstream tasks.

One way to glimpse into the pretraining black box is to observe the performance of downstream language tasks on checkpoints produced during the pretraining. Such

checkpoints were produced at every fifth epoch for at total of 32 saved models. At every checkpoint, fine-tuning and evaluation on DaNE was performed using the hyperparameters from Table 6.2. The result is shown in Figure 6.2.

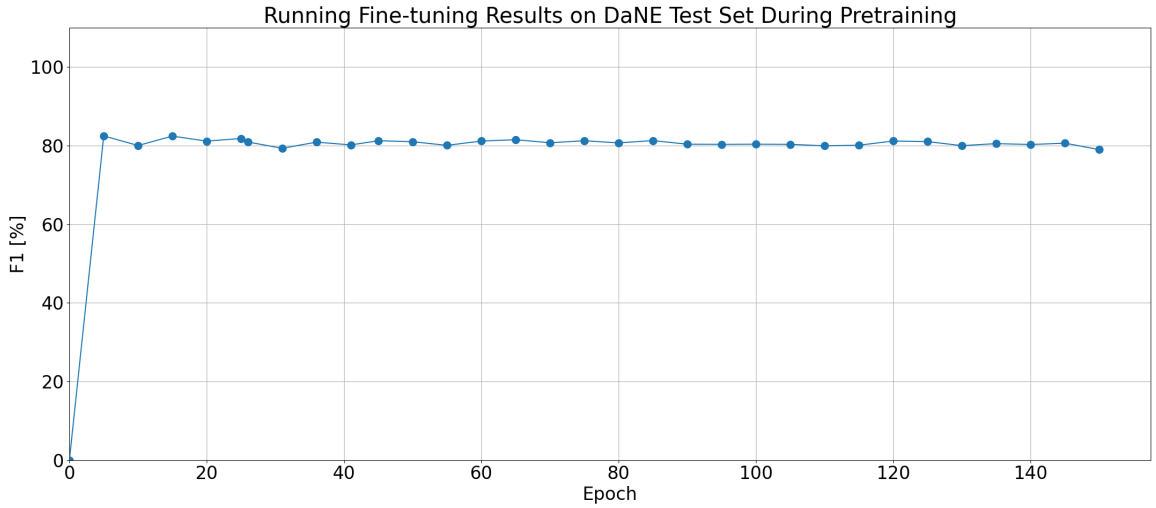

Figure 6.2: Fine-tuning results on DaNE using different pretraining checkpoints.

This result of 0 correct predictions at epoch 0 shows that pretraining is necessary to get good performance on NER. Somewhat surprisingly, however, after five epochs – or maybe even less – additional pretraining does not improve downstream results. In fact, the curve almost perfectly mirrors the MLM accuracy from Figure 5.2, indicating that the model primarily relies on its BERT subset for NER.

It should be noted that NER is just one downstream task and that other such tasks may see more benefit from additional pretraining. However, another explanation may be more probable: The Danish Wikipedia is so small that it only takes a few epochs for DaLUKE to learn what it can from it – especially given that Wikipedia made up part of da-BERT's training data, so it already has been seen [Bot19]. This would also help to explain the much faster convergence of word accuracy compared to entity accuracy. The pretraining dataset only has a little over 100 million subword tokens (see Table 3.1) and 7.2 million entity annotations – a small number for a model with 270 million parameters and an architecture with a well-known near insatiable thirst for data.

6.1.3 Baseline

To set up a baseline model, the main model is retrained using only 50 epochs, but otherwise using the same hyperparameters, shown in Table 4.3. This allows for a comparison to ablation studies where compute resources did not allow a full 150 epoch pretraining.

The final results of this pretraining are summarized in Table 6.1 with the accuracy development shown on Figure 6.3.

Model	Top k accuracy [%]	$k = 1$	$k = 3$	$k = 5$	$k = 10$	$k = 25$	$k = 50$
Baseline	Masked words	24.2	30.9	34.4	39.7	47.7	53.6
	Masked entities	28.8	39.0	43.8	50.6	59.6	66.3

Table 6.1: The top k accuracy of the baseline model in the 50'th and last epoch.

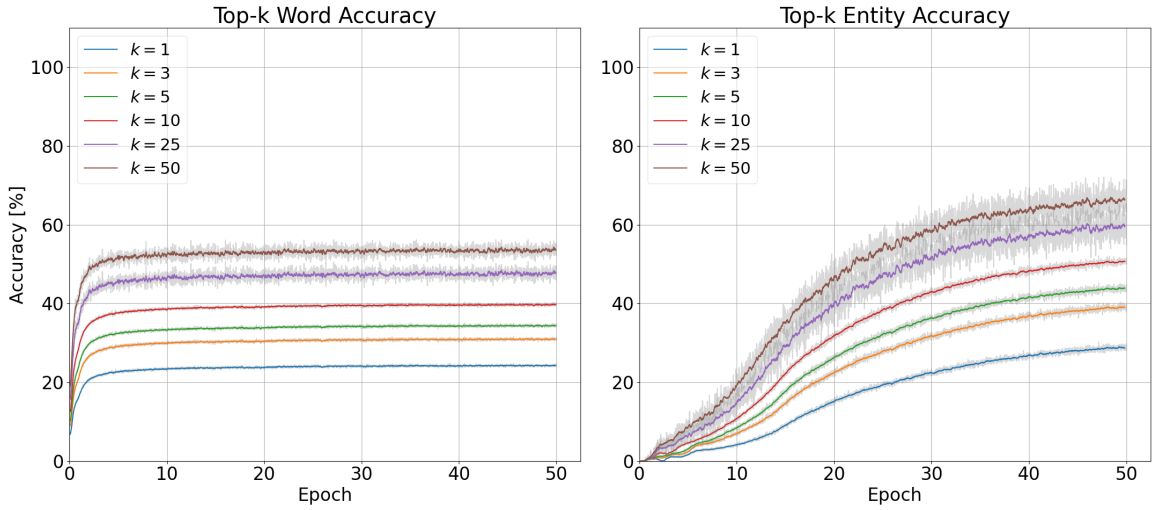

Figure 6.3: Baseline masked language and masked entity accuracy throughout pretraining. All curves are smoothed using a rolling average.

The model was subsequently fine-tuned for NER on DaNE following the approach described in Section 4.2.2 and the following hyperparameters were non-systematically selected:

Parameter	Value
Epochs	10
Batch size	16
Peak learning rate	$2 \cdot 10^{-5}$
LR warmup steps proportion	6 %
Dropout (pretrained model)	0.1
Dropout (final, linear layer)	0.025
Weight decay	0.05
Loss weighting	No

Table 6.2: Hyperparameters for baseline fine-tuning experiment and the following ablation experiments.

This resulted in the following performance on the DaNE test dataset.

Model	F1 [%]					Precision [%]	Recall [%]
	Avg.	LOC	PER	ORG	MISC	Avg.	Avg.
Baseline	82.4 ±3	89.5	92.9	74.4	68.8	86.7	78.4

Table 6.3: The baseline fine-tuning results

The results of the baseline experiment along with the following experiments are shown together in a final overview at Section 6.1.8.

6.1.4 Entity-aware Self-attention

The entity-aware self-attention mechanism, one of Yamada et al.'s key contributions to the transformer, was used for our pretraining task. An experiment using the traditional attention, which does not discriminate between word and entity tokens, was performed.

The produced model yields worse results, shown at Table **??**, than the baseline model.

Model	Top k-accuracy [%]	$k = 1$	$k = 3$	$k = 5$	$k = 10$	$k = 25$	$k = 50$
BERT attention	Masked words	20.6	26.3	29.4	34.6	42.4	48.7
	Masked entities	12.3	21.0	25.7	32.8	42.9	51.5

Table 6.4: The top k accuracy of the pretrained model trained without entity-aware self-attention in the 50'th and last epoch.

After fine-tuning, the following results were achieved:

Model	F1 [%]					Precision [%]	Recall [%]
	Avg.	LOC	PER	ORG	MISC	Avg.	Avg.
BERT attention	79.6 ±3	85.8	92.8	70.9	64.8	84.5	75.2

Table 6.5: The fine-tuning results of the traditional attention experiment.

From the lower NER and masked language performances, it is concluded that this mechanism explicitly modeling the difference between the two token domains is an important part of the DaLUKE results.

6.1.5 Dataset Augmentation

A key addition to the pretraining pipeline was the addition of extra entity annotations not already in the Wikipedia articles themselves using pattern matching as explained in Section 3.1.1. This resulted in a 47 % growth in the number of annotations as shown in Table 3.1. It was argued that this should improve performance which is checked by pretraining a model with the original, un-altered data.

The pretraining terminated to the following performance.

Model	Top k-accuracy [%]	$k = 1$	$k = 3$	$k = 5$	$k = 10$	$k = 25$	$k = 50$
No data aug.	Masked words	25.3	32.2	35.6	40.9	48.8	54.6
	Masked entities	34.0	42.9	47.2	53.5	62.3	68.8

Table 6.6: The top k accuracy of the pretrained model trained without dataset augmentation in the 50'th and last epoch.

At a first glance, these masked language modeling results, strictly dominating the numbers for the baseline, seem to suggest that the dataset augmentation was ill-advised. Apart from the obvious problem of judging a model by its performance on the training dataset, there is a large issue with the metric in this case: The dataset augmentation changed the benchmark itself as the masking task of the baseline model also includes automatically annotated, and thus somewhat dubious, entity links. This can explain the clear gain in entity masking performance of this experiment. Whether this reason has any merit in explaining the increased word accuracy found in this ablation study is less clear to us and relates to the model synergy effects in this joint task.

As a more unbiased benchmark, this model was fine-tuned on the DaNE dataset using the hyperparameters at Table 6.2 resulting in the following performance:

Model	F1 [%]					Precision [%]	Recall [%]
	Avg.	LOC	PER	ORG	MISC	Avg.	Avg.
No data aug.	83.0 ±3	84.4	95.1	76.4	72.1	85.4	80.8

Table 6.7: The fine-tuning results of the dataset augmentation pretraining experiment.

The ablation study gets a substantially higher recall than the baseline resulting in a higher micro average F1 score on the NER task, though the baseline outperforms in precision. All in all, these benchmarks cannot be used to support our theorized problem of false negatives in the volunteer-produced annotations, or at least not that it is mitigated by this augmentation approach.

As the addition of these 47 % annotations, which can be called bronze standard, did not directly stop the learning, we still propose such automated annotation as an avenue for further development of DaLUKE.

6.1.6 Impact of Danish BERT

As described in section 2.5, many of the weights are initialized from a base transformer. This transfer learning method ideally gives DaLUKE all the contextual language understanding of da-BERT. To test this hypothesis, a pretraining is conducted without initializing the weights to da-BERT.

Model	Top k accuracy [%]	$k = 1$	$k = 3$	$k = 5$	$k = 10$	$k = 25$	$k = 50$
No transfer	Masked words	47.9	59.3	63.9	69.8	76.5	81.0
learning	Masked entities	81.7	87.2	88.9	90.9	93.2	94.6

Table 6.8: The top K accuracy of the pretrained model trained without initializing weights from da-BERT in the 50'th and last epoch.

It is immediately clear from Table 6.8 that this model performs noticeably better at both the masked word and masked entity tasks than the other models so far. This is despite starting from 0 in the masked word task (see Figure 6.4). Because of this, good NER performance could reasonably be expected. This did not happen - on the contrary, results were much worse than the baseline.

Model	F1 [%]					Precision [%]	Recall [%]
	Avg.	LOC	PER	ORG	MISC	Avg.	Avg.
No transfer learning	71.0 ±4	79.0	83.5	55.4	61.7	76.9	65.9

Table 6.9: The fine-tuning results of the experiment where weights were trained from scratch.

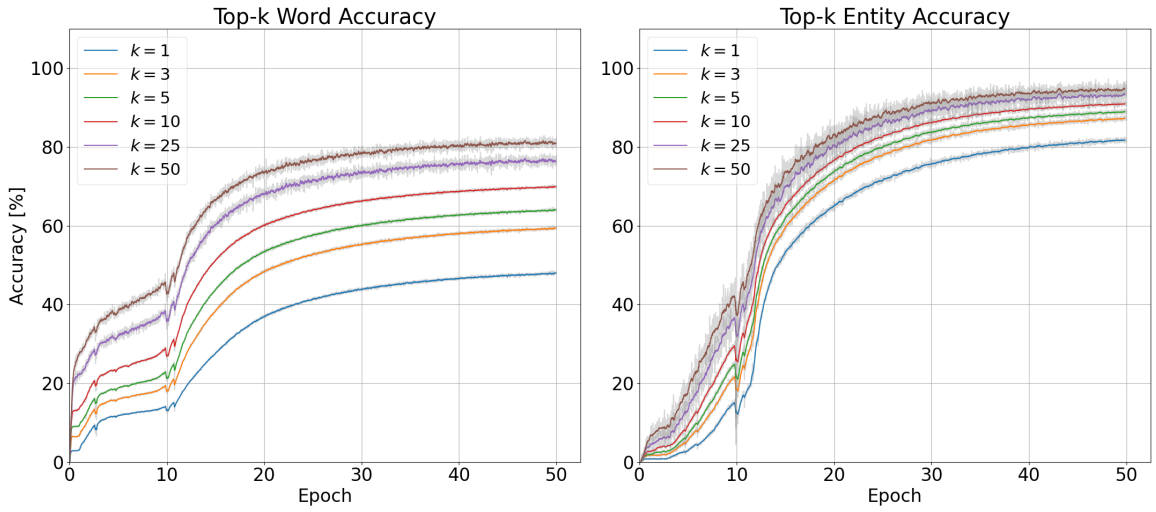

Figure 6.4: Development of the masked word and entity accuracies during pretraining without initializing from da-BERT weights. All curves are smoothed using a rolling average.

This result is unintuitive in that a model scoring highly on *both* masking tasks can underperform on NER. While the running fine-tuning presented in Section 6.1.2 showed decoupling between entity masking and NER results, this result further raises the issue of the model overfitting on the limited dataset even when the masking of words and entities is dynamic.

Thus, the transfer learning from da-BERT turns out to be a help here. From Figure 6.4, the initialization seems to provide two things other than initial MLM accuracy:

- A regularization effect that prevents the model from overfitting to the dataset.
- Increased pretraining stability. The accuracy curves contain multiple of sudden jumps and drops that are much rarer in other pretrainings.

This stabilization effect is supported by the stable distribution of parameters inherited from da-BERT shown at Figures 6.1. It should be noted that because no weights came from da-BERT, no weights were ever locked throughout the pretraining in this experiment. A similar experiment with weights unlocked and initialization from da-BERT showed pretraining accuracies much closer to the baseline though with somewhat higher entity prediction accuracy, indicating that the regularizing effect is indeed the transfer learning from da-BERT. The details of this extra experiment are listed in Appendix A.1.

6.1.7 Entity Vocabulary Size

For the English LUKE, Yamada et al. used an entity vocabulary of the $500 \cdot 10^3$ entities [Yam+20, Sec. 3.4] even though the English Wikipedia contains $\sim 6 \cdot 10^6$ content pages[6.1]. For the Danish Wikipedia, however, the number of content pages[6.2]is $\sim 267 \cdot 10^3$ resulting in the DaLUKE entity vocabulary containing $\sim 225 \cdot 10^3$ entities. As this is a much smaller world of entities, the entity vocabulary of the main DaLUKE model was not cut by frequency.

An experiment was performed where entities mentioned less than 50 times were excluded from the data, resulting in a vocabulary of 19,119 entities. This was done on the augmented data explained in Section 3.1. After pretraining for 50 epochs, the following performance was observed.

Model	Top k accuracy [%]	$k=1$	$k=3$	$k=5$	$k=10$	$k=25$	$k=50$
Limited entity vocab.	Masked words	23.9	30.5	33.9	39.2	47.1	53.0
	Masked entities	30.6	42.2	48.9	55.8	65.9	66.3

Table 6.10: The top k accuracy of the entity vocabulary-limited model in the 50'th and last epoch.

The performance on the word prediction task is slightly worse for this entity vocabulary experiment, while higher scores are seen on the masked entity task. Like the data augmentation experiment, attention must again be put on the changes in the benchmark itself; the accuracy of 31 % is calculated in a $\sim 20 \cdot 10^3$ class problem while the baseline accuracy of 29 % is calculated over almost ten times as many classes.

[6.1]English Wikipedia Statistics: `https://en.wikipedia.org/wiki/Special:Statistics`. Visited March 3, 2021.

[6.2]Danish Wikipedia Statistics: `https://da.wikipedia.org/wiki/Speciel:Statistik` Visited March 3, 2021.

The NER performance of this model was, following the approach in the previous experiments, found to be close to the performance of the baseline with an average F1 0.2 % points worse.

Model	F1 [%]					Precision [%]	Recall [%]
	Avg.	LOC	PER	ORG	MISC	Avg.	Avg.
Limited entity vocab.	82.2 ±3	89.6	93.5	74.0	68.7	85.0	79.5

Table 6.11: The fine-tuning results of the entity vocabulary limiting pretraining experiment.

Conclusively, the results of this experiment are close to the baseline, signifying robustness in the approach. However, with these hyperparameters, the removal of entities does not seem beneficial for the Danish dataset.

This experimental filtering of the DaLUKE entities preserves 7 % of the content pages, while 8 % were preserved for the English LUKE. Intuitively, it would make sense that the same proportional filtering works better in English, as the absolute number of entities making up the implicit knowledge base of the model might be important: Smaller datasets, corresponding to smaller language areas, do not equate smaller shared worlds of named entities.

6.1.8 Summary

Table 6.12 and 6.13 show the masked language and entity modeling results and fine-tuning results, respectively, of all the just discussed experiments for comparison.

Model	Top k-accuracy [%]	$k = 1$	$k = 3$	$k = 5$	$k = 10$	$k = 25$	$k = 50$
Baseline	Masked words	24.2	30.9	34.4	39.7	47.7	53.6
	Masked entities	28.8	39.0	43.8	50.6	59.6	66.3
BERT attention	Masked words	20.6	26.3	29.4	34.6	42.4	48.7
	Masked entities	12.3	21.0	25.7	32.8	42.9	51.5
No data aug.	Masked words	25.3	32.2	35.6	40.9	48.8	54.6
	Masked entities	34.0	42.9	47.2	53.5	62.3	68.8
No transfer learning	Masked words	**47.9**	**59.3**	**63.92**	**69.8**	**76.5**	**81.0**
	Masked entities	**81.7**	**87.2**	**88.9**	**90.9**	**93.2**	**94.6**
Limited entity vocab.	Masked words	23.9	30.5	33.9	39.2	47.1	53.0
	Masked entities	30.6	42.2	48.9	55.8	65.9	66.3

Table 6.12: Overview of final pretraining results for all the experiments presented previously. The best result for each metric is shown in boldface with second-best result underlined (motivated by the quite boring distribution of bests).

Model	F1 [%]					Precision [%]	Recall [%]
	Avg.	LOC	PER	ORG	MISC	Avg.	Avg.
Baseline	82.4 ±3	89.5	92.9	74.4	68.8	**86.7**	78.4
BERT attention	79.6 ±3	85.8	92.8	70.9	64.8	84.5	75.2
No data augmentation	**83.0** ±3	84.4	**95.1**	**76.4**	**72.1**	85.4	**80.8**
No transfer learning	71.0 ±4	79.5	83.5	55.4	61.7	76.9	65.9
Limited entity vocab.	82.2 ±3	**89.6**	93.5	74.00	68.7	85.0	79.5

Table 6.13: Overview of the pretraining experiment fine-tuning results presented in the previous sections. Many of the differences are small and within the 95 % finite sample margin of error, so the performance must be judged in conjunction with masked task performance.

6.2 Fine-tuning Performance

As with pretraining, fine-tuning has interesting elements that are worth investigating further. For these experiments, the main pretrained model from Section 5.2 are used but with the optimized fine-tuning hyperparameters from Table 4.4. Following the pretraining experiments, all results are reported with MISC.

6.2.1 Stability of LUKE Fine-tuning

Reproducibility of English LUKE For the reproduction of the English LUKE, our results were initially underwhelming as five repetitions of the fine-tuning of LUKE large on CoNLL-2003 consistently slightly underperformed the reported result of 94.3% micro average as seen at Table 6.14. This disappointment was backed up by an approximate 95 % confidence interval on the true mean of the avg. F1 score being marginally below the reported value[6.3].

After investigation, this, to us, somewhat surprising result was revealed to be overturned when fine-tuning with Python version 3.6 and PyTorch 1.2 instead of using newer versions as done in the initial run. Using these correct versions, the same as Yamada et al. reported in their package file, our results were much closer to the reported numbers as shown in Section 5.1.

[6.3]Using the one-sample Student t-based confidence interval relying on an unchecked normality assumption on means, [Bro+18, Method 3.9] our five repetitions yield a 95 % confidence interval of $[92.53\,\%, 94.26\,\%]$. To conclude a significant difference, more repetitions are necessary, but we were initially worried by this result.

All in all, the slight hiccups in reproducing the fine-tuning results show the difficulties in empiric evaluations of deep methods strongly relying on indeterminism – especially as the underlying framework, PyTorch, does not guarantee reproducible results across versions and platforms, even if using identical seeds for the random number generator [Conc].

Model	Over 5 repetitions	Micro avg.	LOC	PER	ORG	MISC
LUKE large	Mean F1 [%]	93.4	95.2	97.2	93.5	85.2
	Std. [%]	0.07	0.2	0.08	0.2	0.4

Table 6.14: Results when pretraining and evaluating LUKE with Python version 3.8 and PyTorch 1.4. Using these software versions, the model seemingly underperforms the reported accuracy.

Random number generation variance The same repetition of the fine-tuning procedure carried out for English LUKE is done for the main DaLUKE model by training with five different random number generation (RNG) seeds. The results are shown in Table 6.15.

Seed	F1 [%]					Precision [%]	Recall [%]
	Avg.	LOC	PER	ORG	MISC	Avg.	Avg.
1	83.7	85.8	94.3	78.0	72.3	88.7	79.2
2	81.5	84.3	92.2	71.9	74.7	83.9	79.3
3	84.5	88.6	93.8	78.0	75.1	86.9	82.2
4	81.3	82.4	93.1	72.8	72.8	83.8	79.0
5	82.4	84.9	93.2	74.6	74.0	84.9	80.1
Mean	82.7	85.2	93.3	75.0	73.8	85.6	80.0
Std.	1.4	2.3	0.8	2.8	1.2	2.1	1.3

Table 6.15: Repeating the fine-tuning procedure of the main model, reporting results.

Unfortunately, the standard deviation is rather large compared to the margins between many of the experiments. For reference, seed 1 was used for all experiments bar this one. The stochasticity of the fine-tuning procedure, possibly driven by the limited size of the training set, casts doubt on NER comparisons. As Table 6.15 shows, one of the models (seed 3) even achieves SOTA.

Dataset split variance Another way to look at fine-tuning indeterminism is the randomness in the dataset. Predefined splits, chosen randomly by the dataset creators

at the Alexandra Institute [Hvi+20], into training, development and testing datasets were used for all experiments. A less noisy estimate of generalization performance is generally obtained by K-fold cross-validation as opposed to this simple hold-out testing method [Bis06, Sec. 1.3]. This is performed using $K = 5$ random splits as this gives a training set of approximately the same size as for the predefined splits, that is, 4383 examples in the DaNE splits and 4410 in cross-validation training. The cross-validation test sets will thus be larger, as no development set is used.

Split	F1 [%]					Precision [%]	Recall [%]
	Avg.	LOC	PER	ORG	MISC	Avg.	Avg.
1	89.1	91.5	95.1	82.5	85.1	89.8	88.5
2	86.1	87.5	94.2	76.0	82.2	88.2	88.1
3	86.9	89.7	93.5	76.7	84.5	88.3	85.5
4	87.4	90.6	94.6	81.2	79.2	87.8	87.1
5	87.6	90.9	95.3	79.0	86.9	88.1	87.1
Mean	87.4	90.0	94.5	79.1	83.6	88.4	87.3
Std.	1.1	1.6	0.7	2.8	3.0	0.8	1.7

Table 6.16: 5-fold cross-validation results over the DaNE dataset. The training, dev. and test sets were all combined and then split in five different ways with each split having 80 % training data and 20 % test data, on which the F1 score is calculated.

These results at Table 6.16 imply that the DaNE testing set performance of 82.89 % understates the accuracy of this method. An explanation can be that the small testing set (565 sentences) simply contains more difficult examples than other parts of the dataset. This theory is substantiated by the differing distributions of entities between DaNE splits (testing contains fewer LOC's and more ORG's, see Figure 3.1) and that we consistently observed higher performance on the development set than on the testing (such as in the hyperparameter optimization results at Table A.3).

It must be noted that we cannot claim to present an unbiased estimate of the generalization performance, even on datasets following the same distribution, as the main experiment hyperparameters, that were also used here, were chosen based on performance on much of the data.

Dataset generalization ability It was noted in the results section that DaLUKE appears to generalize well to the two other Danish NER datasets. To add detail to the ability to generalize across these three datasets, DaLUKE is also fine-tuned for the two other datasets and the results for each model are reported across datasets at Table 6.17.

Test set	Model trained on	Micro Avg. [%]				Class F1 [%]			
		F1	Prec.	Rec.	F1 $_{\not\text{MISC}}$	LOC	PER	ORG	MISC
DaNE	DaNE	82.8	84.6	81.1	85.1	87.0	94.1	73.1	74.5
DaNE	Plank	69.0	60.5	80.2	78.8	81.6	92.5	56.0	13.2
DaNE	WikiANN	–	–	–	66.8	73.4	74.8	44.1	–
Plank	DaNE	63.4	80.8	52.1	73.1	76.0	90.4	45.6	4.6
Plank	Plank	74.9	78.0	72.0	74.9	76.2	90.3	50.5	41.0
Plank	WikiANN	–	–	–	58.4	62.8	74.6	21.3	–
WikiANN	DaNE	–	–	–	67.8	67.9	87.5	51.2	–
WikiANN	Plank	–	–	–	60.5	68.5	88.0	29.2	–
WikiANN	WikiANN	–	–	–	93.4	93.4	94.7	88.3	–

Table 6.17: Results when fine-tuning and evaluating across datasets. Results without the MISC categories are included as WikiANN does not include this category.

As the table shows, generalizability is not a given. While some LOC and PER generalize fairly well between datasets, ORG, and especially MISC, see very poor performance when evaluated on another dataset than the model was fine-tuned on. One cause for this may be the different distributions of labels between datasets. Another is the understanding the entity category definitions. WikiANN is automatically annotated and so is obviously different from the DaNE and Plank, where human language understanding directly played a role. However, as DaNE and Plank have different annotations, clearly the authors had different interpretations of what the categories meant (as is especially clear with the MISC category), even though both used the CoNLL-2003 guidelines. These issues cast doubt on the generalizability of NER. [Hvi+20; Pla19; Pan+17; TD03]

6.2.2 Class-weighted Loss

Cross entropy loss weighted by the number of training examples in each class was speculated to improve performance on the class-imbalanced NER task in which 99.3 % of DaNE spans are not entities. In the hyperparameter search, all combinations of learning hyperparameters were also tried with this weighting of loss defined in eq. (4.4). From the results at Table A.3, no general improvement could be seen when weighting, and the best combinations were slightly worse when using this approach.

Two experiments with suboptimal hyperparameters of high learning rate, high dropout, and low weight decay resulted in the model degenerating to predicting the positive entity classes much too often as seen from Table 6.18. Interestingly, this problem was mitigated when using class-weighted loss, an unintuitive result in our estimation,

as we expected this weighting to motivate more positive predictions.

Batch size	Learning rate	Weight decay	Dropout	Loss weight	F1	Precision	Recall
8	$5{\cdot}10^{-5}$	0.01	0.1	Yes	84.9	80.4	90.0
8	$5{\cdot}10^{-5}$	0.01	0.1	No	31.5	19.8	76.4
16	$5{\cdot}10^{-5}$	0.01	0.1	Yes	82.6	76.8	89.3
16	$5{\cdot}10^{-5}$	0.01	0.1	No	29.2	18.0	76.0

Table 6.18: The only hyperparameter search experiments in which class-weighted loss improved performance. The unweighted experiments suffer from very high recall and low precision, corresponding to many false positives.

Conclusively, this idea did not benefit our main results, a testament to the robustness of gradient learning, but gave an example of how strategies for class imbalance can be necessary when the learning is less stable.

6.2.3 Feature Usage

An entity candidate forward passed through the final, classifying layer in the NER model consists of the concatenation of word representations of the first and final subword tokens, and, novelly for LUKE, a contextual representation of the entity span. Two fine-tuning experiments are performed by altering this entity feature approach by observing the performance when the subword token and entity representations are used individually.

Model	F1 [%]					Precision [%]	Recall [%]
	Avg.	LOC	PER	ORG	MISC	Avg.	Avg.
Subword tokens	81.2 ±3	**87.6**	92.4	70.9	71.6	84.6	78.1
Entity tokens	81.7 ±3	85.1	93.2	71.4	74.0	**85.0**	78.6
Both (default)	**82.8**±3	87.0	**94.1**	**73.1**	**74.5**	84.6	**81.1**

Table 6.19: Fine-tuning results using different features for the classifier. Best results are marked in bold.

From the results shown at Table 6.19, it is learned that the model can perform the NER task with high performance using any of the three feature combinations. This is explainable by both entities and words interacting with each others' representations in the transformer and thus affecting the final representations given to the classifier no matter how these are chosen.

The experiment where only the entity tokens are used is arguably a simple and more elegant NER approach as the selection of the first and last subword tokens to describe a span seems arbitrary. These results make such an approach seem viable.

6.3 Predictions: What Is Learned?

Until this point, we have used the benchmark performance to judge the models. To get a better understanding of the model, this section includes attempts to introspectively interpret the model predictions.

6.3.1 Masked Language Predictions

To demonstrate DaLUKE's ability to consider context, an example, and some variations on it, are constructed. The example is given below with text sequence first, followed by entity annotations.

- "[MASK] blev angrebet af USA i 2003 som følge af terrorangrebet den 11. september 2001. Saddam Hussein blev fanget i december."
  Truth: "Irak"
- { *[MASK]* at word #1,
  USA at word #5,
  2003 at word #6,
  Terrorangrebet den 11. september 2001 at words #11-15,
  Saddam Hussein at words #12-13 }
  Truth: "Irak".

First consider the shorter version: "[MASK] blev angrebet af USA i 2003 som følge af terrorangrebet d. 11. september 2001." Given this text piece, DaLUKE predicts "afghanistan"[6.4] with a certainty of 15.95 %. While certainly a reasonable guess, it is incorrect, as Afghanistan was attacked in 2001. For context, "irak" is the second guess of the model with 4.98 % certainty. Furthermore, the masked "Irak" entity is predicted as "USA", but only with 7.61 % certainty. Adding the additional context of Saddam Hussein being captured, however, makes DaLUKE predict "irak" with 17.64% certainty. Furthermore, the masked "Irak" entity is predicted as "Irak" with 15.57 % certainty.

[6.4] Model predictions are lower-cased, as the da-BERT tokenizer is lower-cased.

This example highlights how DaLUKE takes advantage of context to substantially improve its predictions. The following examples show similar results. For simplicity, no entities are annotated.

- "I [MASK] spises der meget and." – "danmark" is predicted, which makes perfect sense but was not "december" as was first intended with the sentence.
  "I [MASK] spises der meget and. Det er nemlig julemåneden." – "december" is now predicted.
- "Danmarks vigtigste lov er [MASK]." – "bekendtgørelser", which, while grammatically correct, does not make much sense in the context.
  "Danmarks vigtigste lov er [MASK]. Den blev underskrevet i 1849." – "grundloven" is correctly predicted.

6.3.2 DaLUKE Representations: The NER Geometry

DaLUKE produces contextualized representations of both words and entities and these were, for the NER task, successfully adapted to show a level of language understanding. These abstract representations which, ideally, condense language understanding into 768 reals are here subjected to examination. The entity span representations, the concatenation of contextual word and entity representations, $\mathbf{v}_i \in \mathbb{R}^{2304}$, are analysed by conducting DaLUKE inference on every possible span in the DaNE training data, and saving each result, thus producing a sequence of representations $\mathbf{v}_{1,\ldots,975312}$. The pretrained, but not fine-tuned model was used. Dimensionality reduction for visualization on the representations was performed using principal component analysis (PCA), the t-SNE (t-distributed stochastic neighbour embedding) algorithm [MH08], and the UMAP (uniform manifold approximation and projection for dimensionality reduction) algorithm [MHM20].

Implementation-wise, Sci-Kit Learn was used for t-SNE [Ped+11] and UMAP Learn was used for UMAP [McI+18] with hyperparameters shown at Table 6.20. The set of 975,312 entity spans in the DaNE training set was randomly subsampled to 100K examples for these non-linear dimensionality reduction algorithms due to compute limitations. Even with this limit and with hyperparameters chosen to favour global structure, t-SNE and UMAP results were difficult to interpret, see Appendix B.1, motivating visualizing the dataset where only the 4,003 entity spans that corresponded to true entity annotations were included.

Parameter	Value
t-SNE perplexity	1,000
UMAP N neighbours	300
When only including positives	
t-SNE perplexity	100
UMAP N neighbours	50

Table 6.20: The hyperparameters used for the dimensionality reduction algorithms run on the DaLUKE entity representations. All other hyperparameters followed the defaults in Sci-Kit Learn 0.24.1 and UMAP Learn 0.5.1.

Dimensionality reduction on the full dataset yields weakly structured visualizations of the geometry, mostly dominated by giant clusters, as seen for PCA at Figure 6.5, which also shows some clustering of true entities, also seen for the others in Appendix B.1. The true, underlying dimensionality of the representations also seems high as the first 100 principal components only explain less than 60 % of data variance, see Figure B.3, an indication of well-learned representations which, ideally, should exploit all vector coordinates by having minimal latent dimension covariance.

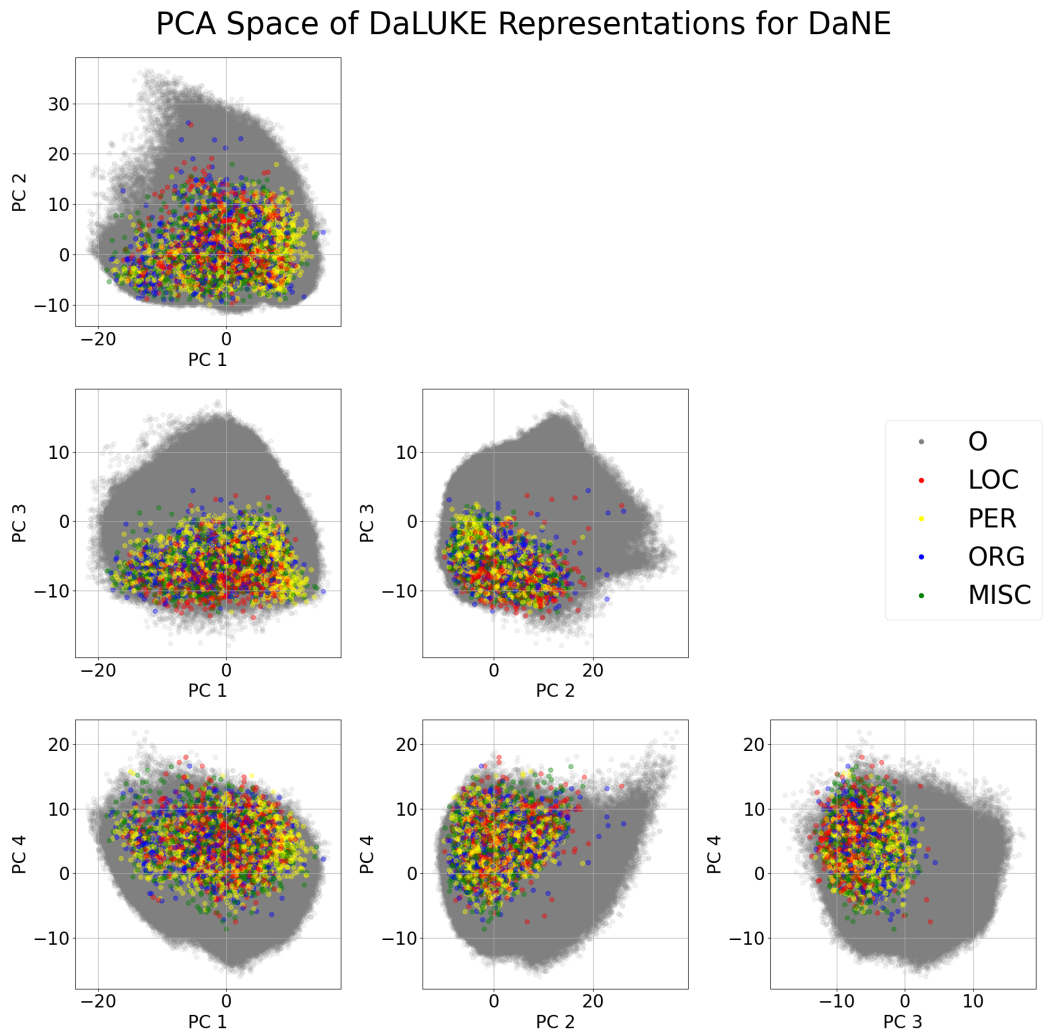

Figure 6.5: The first four components visualized against each other from PCA performed on all 975,312 possible entity spans in the DaNE training data. Spans in grey are annotated entities in the dataset. The non-entity spans are more dispersed, intuitively making sense when considering that these include all possible spans of words only limited by the maximum length of 16 subword tokens.

For the reductions in the positive label-only dataset, even more meaningful structure is apparent with all three methods, at Figures 6.6, 6.7, 6.8, showing slight grouping of entities in the same NER class, indicating that the pretrained model, even before fine-tuning, represents language in a way that is close to the human-annotated dataset.

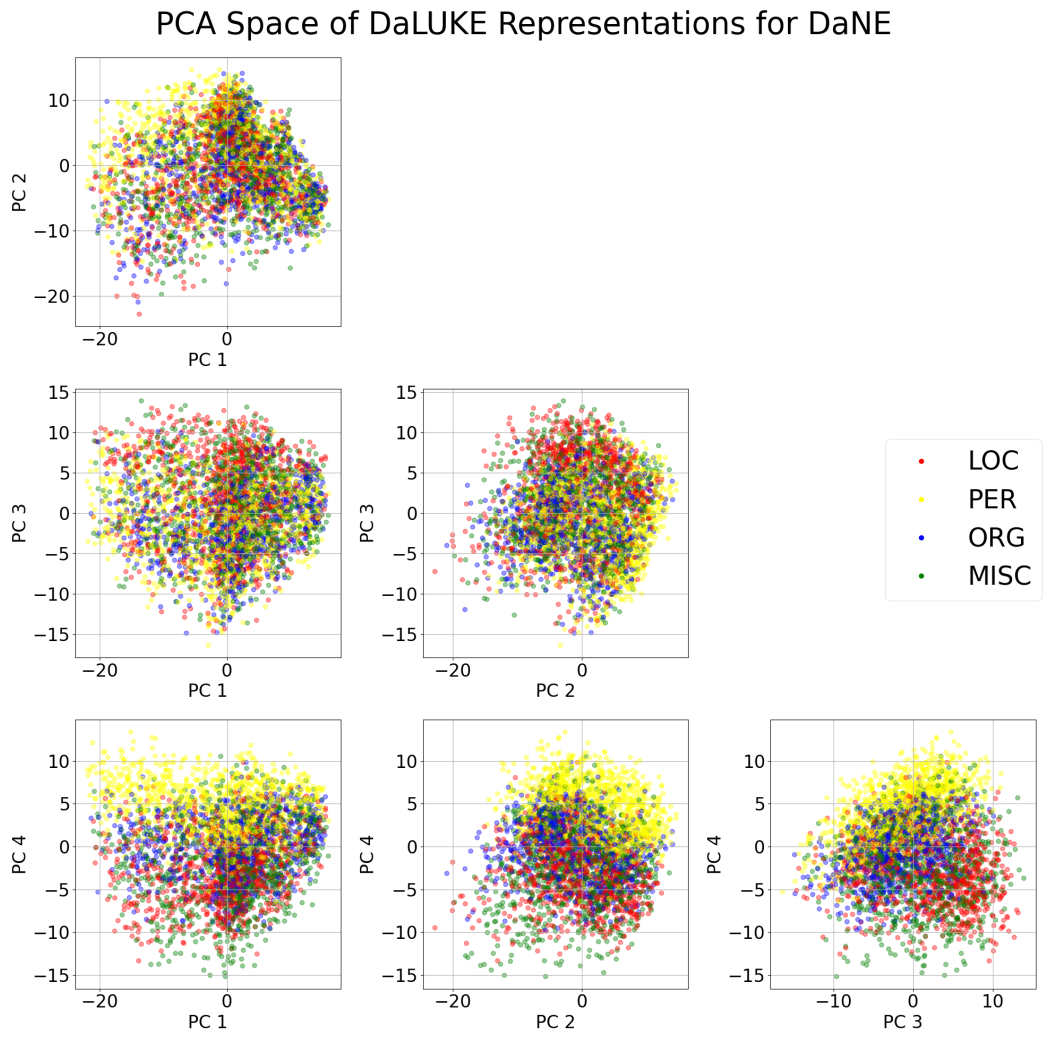

Figure 6.6: The first four components visualized against each other from PCA performed on only the 4,003 entity spans that are annotated to a NE class in the DaNE training data. Class separation is observed in multiple of the components, exemplifying why the NER classification problem is solved well by DaLUKE which only requires few epochs to distinguish classes.

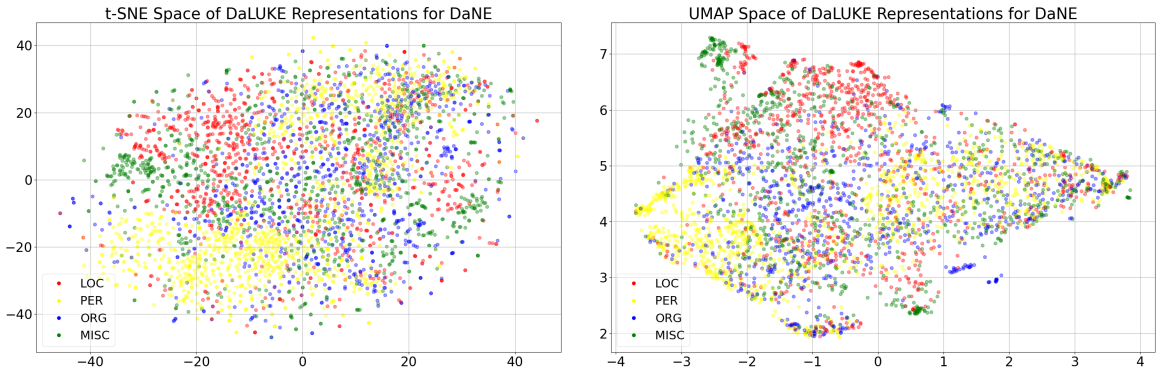

Figure 6.7: *t*-SNE and UMAP projections of the DaNE dataset containing entity spans with positive labels. While the classes are not strongly separated, some clusters consisting of mostly one class are clearly visible in both projections.

The obvious question is what these plots mean: What do the clusters and the dimensions themselves correspond to? This is far from a trivial task: While it is possible to correlate the dimensionality reduced coordinates to the original dimensions, more clever thinking is required to correlate both to human language understanding. We mostly leave this task for further work but perform a case-based inspection of the predictions. Our qualitative observations are summarized below with supporting examples in Table 6.21.

- The second principal component in the dataset including all data seem to express something related to span position, as the highest values in this dimension almost all correspond to examples with the span at the very end of the sentence. Based on this, it makes sense that the plots in Figure 6.5 show the actual entities having moderate values.
- Low values of the fourth principal component in the positive-label only dataset seem to correspond to a very specific type of words: Adjectivizations of political, geographic regions such as "Danish" or "English". As explained in Section 3.2.3, such entities should have the label MISC. This observation fits with Figure 6.6 showing MISC generally having the lowest 4th dimension values and LOC coming in second.
- For *t*-SNE on the positive-only dataset, no such clear dependencies on coordinate numeric values were observed, explainable by *t*-SNE not returning linear projections, but an approximate recreation of data point distances. Two examples very

close in t-SNE distances are shown in Table 6.21; these are also semantically very similar in entity type and role.

- For UMAP fitted to the examples with positive labels, the clear cluster placed around $(-2.5, 7)$ is examined. Here, every entity is revealed to a derivative of the word *Danmark*. Some of these are MISC (adjectivizations, demonyms) and some LOC (*Danmark* itself) as seen at Figure 6.7.

Data	Alg.	Vector	Class	Example
Full	PCA	$\begin{bmatrix} -11.75 \\ \underline{36.44} \\ -4.85 \\ 16.42 \\ \vdots \end{bmatrix}$	O	De følgende 10 år er der ydet omkring 800 mio. kr. til enkeltprojekter **og forskningsinstitutter**.
Full	PCA	$\begin{bmatrix} -8.78 \\ \underline{32.14} \\ -1.64 \\ 14.64 \\ \vdots \end{bmatrix}$	LOC	Den syriske leder ankom i går til Abu Dhabi efter besøg i Saudi-Arabien og **Kuwait**.
Positives	PCA	$\begin{bmatrix} -0.23 \\ -5.63 \\ -9.60 \\ \underline{-15.17} \\ \vdots \end{bmatrix}$	MISC	Det **jugoslaviske** præsidentråd appellerede i sidste øjeblik til FN om at undlade at iværksætte en boykot og opfordrede til, at der i stedet indkaldes til en international konference om konflikten.
Positives	PCA	$\begin{bmatrix} 2.30 \\ 5.32 \\ 4.91 \\ \underline{-13.13} \\ \vdots \end{bmatrix}$	MISC	Med indsættelse af **europæiske** fartøjer rykker Vestunionen for første gang i centrum af europæisk sikkerhed efter mange års debat om at lette USAs byrder ved forsvaret af Europa.
Positives	t-SNE	$\begin{bmatrix} \underline{-37.77} \\ -22.41 \end{bmatrix}$	PER	"Piloten forsøgte at rette maskinen op – så kunne jeg ikke se mere, men pludselig var der gnister i luften," siger øjenvidnet **Peter de Neef**.
Positives	t-SNE	$\begin{bmatrix} \underline{-37.75} \\ -22.36 \end{bmatrix}$	PER	"Løfterne om bonus har jeg heller aldrig fået svar på, hvor bare en undskyldning kunne have gjort underværker," understreger **Peter Freil**.
Positives	UMAP	$\begin{bmatrix} -2.54 \\ \underline{7.27} \end{bmatrix}$	MISC	Datoen for den dag i april, da han fik sin tro på det **danske** retssystem tilbage.
Positives	UMAP	$\begin{bmatrix} -2.07 \\ \underline{7.17} \end{bmatrix}$	LOC	Roskilde Domkirke bliver 12. november rammen om den første af en række koncerter, den norske sangerinde Sissel Kyrkjebø giver i **Danmark**.

Table 6.21: Selected entity examples chosen for having high values in resulting dimensions. An example consists of a sentence and an entity candidate – the example entity span is here visualized with boldface words. The coordinate value which is extreme and motivated the inclusion of the example is underlined.

Furthermore, a key quality of the representations is their contextualized nature; this is shown to be present in the reduced dimensions, as multiple of these are correlated with

the length of the example sequence as shown in Figure 6.8.

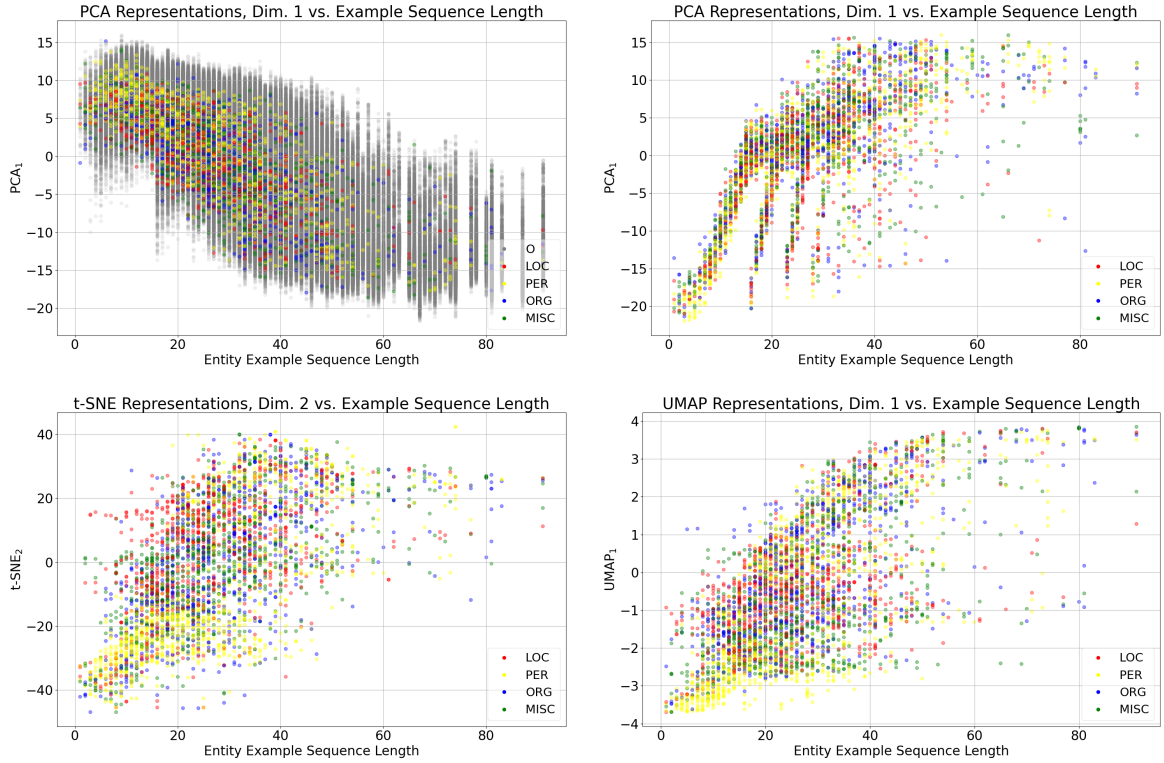

Figure 6.8: Four selected reduced dimensions, the first (top left) from the full dataset and the three others from the dataset only containing positives, visualized as a function of the length of the sentence that the entity example appears in.

6.3.3 When the Model is Wrong

Analysing when the model is right or wrong may provide valuable insight into how the classification decisions are made and how to improve this. For this reason, a confusion matrix (Table 6.22) is constructed for the DaNE test set that shows predicted labels against the true labels[6.5].

[6.5] All confusion matrices in this section are for simplicity and nuance done on the word level and not on entire entity span level. This means that these confusion matrices do not correspond exactly to the reported precision and recall scores, as these were calculated on span level as explained in Section 4.1.1.

		Predicted label				
		LOC	PER	ORG	MISC	O
True label	LOC	91	0	2	3	5
	PER	0	307	2	3	5
	ORG	23	11	154	12	21
	MISC	0	0	16	114	29
	O	5	0	8	14	9,197

Table 6.22: Predicted labels versus the true labels on the DaNE test set. Note that this count is done on word level, so entities spanning multiple words appear multiple times in the table.

From the table, there are two major sources of errors:

1. LOC is often erroneously predicted on ORG entities – but not the other way around.
2. O is often predicted on true labels, especially on ORG and MISC. This results in the relatively low recall of 81.18 % holding back performance compared to the precision of 84.67 %.

Point 1 may have a relatively simple explanation: That many organizations are named after a geographic location while locations more rarely adopt the name of an organization. Consider for instance the organization entity "Bakken" in the (verbatim) test example: "Alle som én er en hyldest til Bakken.".

The model predicts LOC, but while the amusement park, Bakken, would refer to the physical location in another context, this sentence is about an homage to the tradition of the park as an institution, and is as such marked as ORG. While the model is contextual, meaning correctly predicting such entities is possible, examples with as little context as this one requires a degree of very minute real-world knowledge currently not obtained by DaLUKE. This error pattern also occurred with names of theatres, libraries and municipalities.

Text:	Nørrebro	Bibliotek	introducerede	for	et	par	år	siden	NU-bøgerne	...
Truth:	B-ORG	I-ORG	O	O	O	O	O	O	B-MISC	...
Pred.:	B-LOC	I-LOC	O	O	O	O	O	O	O	...

Text:	...	Folkekongressen	skal	give	præsidenten	diktatoriske	beføjelser	.
Truth:	...	B-ORG	O	O	O	B-MISC	O	O
Pred.:	...	O	O	O	O	O	O	O

Text:	...	om	Landsforeningen	Ungbo	har	begået	mandatsvig	...
Truth:	...	O	B-ORG	I-ORG	O	O	O	...
Pred.:	...	O	O	B-ORG	O	O	O	...

Table 6.23: Example sentences where DaLUKE is wrong, including conflating an organization with its location, missing cased and adjective entities, and greedily selecting a subspan instead of the full entity.

Point 2, the prevalence of false negatives (prediction case III, following Table 4.1), can be a calibration problem, but is also, in our estimation, one of the most difficult parts of this dataset, as the definition of NEs starts becoming quite murky. One common cause is the model missing adjectives marked as MISC including "borgerlig", "indremissionsk" and "olympisk". The classification of these as named entities with the MISC label is correct following CoNLL-2003 definition (see Section 3.2.3) but we speculate that this might be a difference in language understanding between English and Danish as this categorization, in our estimation, seems non-intuitive in Danish – in which adjectives are also never capitalized.

In the missed organizations, two possible causes are found. Firstly, using a case-sensitive tokenizer, which the da-BERT one is not [Bot19], might help some errors such as "Markedsudvalget" and "Kulturministeriets". Secondly, cases such as "EFs ministerråd" with the annotation "B-ORG I-ORG" are found. Here, the model only predicts the first word as an organization and gets the entire entity wrong. As "EFs" might follow a pattern more common in the training data than "EFs ministerråd", the former has a higher estimated probability by the model and is greedily selected. This tendency to predict subspans highlights a weakness of the method of greedily selecting partially overlapping spans.

From this qualitative impression of the errors, a subset of which are shown in Table 6.23, no smoking gun was found that problematizes this specific model. Rather, most errors shown here are examples that genuinely are difficult applications of language knowledge. To get more insight into the prediction patterns of DaLUKE specifically, the NER predictions are compared with those of other Danish NER algorithms.

Initially, the similarity of other model predictions and those of DaLUKE are mea-

sured using the proportion of words for which the prediction is the same, shown at Table 6.3.3. Surprisingly, the most similar models are the DaCy models, using the multilingual RoBERTa base and large behind the hood [Ene21], and the multilingual BERT from NERDA, while da-BERT, on which DaLUKE is based, follows behind.

Model	Same prediction as DaLUKE [%]
DaNLP da-BERT	97.86
NERDA m-BERT	98.44
NERDA Ælæctra	97.70
DaCy medium	98.68
DaCy large	**98.76**
DaNLP spaCy	97.65
DaNLP Flair	96.37
Polyglot	93.45
daner	95.90

Table 6.24: Co-prediction frequencies between DaLUKE and other Danish NER algorithms on the DaLUKE testing dataset performed on the word level.

Compared to the da-BERT predictions, the biggest difference is the higher amount of positive predictions as seen at Table 6.3.3. DaNLP da-BERT was not fine-tuned for MISC, so this difference is natural, but DaLUKE also discovers more organizations. DaLUKE also predicts location more rarely and outperforms da-BERT on these (F1: 87.0 % vs. 83.9 %).

		da-BERT predictions				
		LOC	PER	ORG	MISC	O
DaLUKE predictions	LOC	114	0	0	–	5
	PER	0	310	5	–	3
	ORG	10	9	143	–	21
	MISC	1	0	4	–	141
	O	4	3	6	–	9244

Table 6.25: da-BERT predictions vs. DaLUKE predictions.

From examining the examples where the models differ, DaLUKE seems to employ the additional learned knowledge to better disambiguate between similar classes and use context to discover NEs, see Table 6.26.

Text:	...	redegørelsen	i	det	udenrigspolitiske	nævn	...						
Truth:	...	O	O	B-ORG	I-ORG	I-ORG	...						
DaLUKE pred.:	...	O	O	B-ORG	I-ORG	I-ORG	...						
da-BERT pred.:	...	O	O	O	O	O	...						
Text:	"Ih,	hvor	jeg	glæder	mig	til	et	lille	glas,"	lød	det	fra	Lykke
Truth:	O	O	O	O	O	O	O	O	O	O	O	O	B-PER
DaLUKE pred.:	O	O	O	O	O	O	O	O	O	O	O	O	B-PER
da-BERT pred.:	O	O	O	O	O	O	O	O	O	O	O	O	O
Text:	Rapporten	...	er	bestilt	og	betalt	af	Københavns	Amtsråd	...			
Truth:	O	...	O	O	O	O	O	B-ORG	I-ORG	...			
DaLUKE pred.:	O	...	O	O	O	O	O	B-ORG	I-ORG	...			
da-BERT pred.:	O	...	O	O	O	O	O	B-LOC	I-LOC	...			

Table 6.26: Examples of DaLUKE improvements compared to da-BERT include spotting tricky NEs, and correctly identifying correlated. This is possibly accomplished by the knowledge additions allowing the model to use more information from the context.

The same comparison is made to DaCy large which achieves higher performance than DaLUKE – especially driven by improvements in organization and miscellaneous classes. Here, the biggest differences lie in cases where the models have to discern between these two classes and the null class, seen at Table 6.3.3.

		DaCy large predictions				
		LOC	PER	ORG	MISC	O
DaLUKE predictions	LOC	108	0	2	1	8
	PER	0	309	6	3	0
	ORG	9	3	154	5	12
	MISC	0	1	3	120	22
	O	8	4	8	27	9210

Table 6.27: DaCy large predictions vs. DaLUKE predictions.

Examples where DaCy large is right, but DaLUKE wrong, are shown at Table 6.28. The initial explanation of DaCy large's edge over DaLUKE was the use of a BERT large-sized transformer, adding flexibility to the model, theoretically allowing higher-level language features to be learned compared to the base size model used by DaLUKE. From the examples, more reasons are identified. One cause is the use of the case-sensitive byte-pair encoding tokenizer of RoBERTa [Con+20] which we speculate helps to discover more entities, relating to the high recall of DaCy (83.7 % versus the 81.2 % of DaLUKE). Another interesting pattern lies in the strength of the multilingual model used by DaCy correctly predicting entities in foreign languages.

Text:	...	opus	VIII	der	hedder	(continued)				
Truth:	...	B-MISC	I-MISC	O	O					
DaLUKE pred.:	...	B-MISC	I-MISC	O	O					
DaCy pred.:	...	O	O	O	O					
			Il	Cimento	dell'Armonia	e	dell'Invenzione			
			B-MISC	I-MISC	I-MISC	I-MISC	I-MISC			
			O	O	O	O	O			
			B-MISC	I-MISC	I-MISC	I-MISC	I-MISC			

Text:	Der	ligger	fire	skodder	plus	en	hel	Camel
Truth:	O	O	O	O	O	O	O	B-MISC
DaLUKE pred.:	O	O	O	O	O	O	O	O
DaCy pred.:	O	O	O	O	O	O	O	B-MISC

Text:	Den	modtog	han	i	øvrigt	Kulturministeriets	(continued)		
Truth:	O	O	O	O	O	B-ORG			
DaLUKE pred.:	O	O	O	O	O	O			
DaCy pred.:	O	O	O	O	O	B-ORG			
		børnebogspris	for	i	1990				
		O	O	O	O				
		O	O	O	O				
		O	O	O	O				

Table 6.28: Examples from the DaNE test set where DaCy large gives the correct predictions and DaLUKE gives the wrong predictions.

6.4 Future Work

We see promise in this approach of lightly sprinkling some explicit knowledge on top of the model-fitting approach for languages such as Danish. The results are, in our estimation, solid and encouraging. DaLUKE did not, however, sweep away the classic language modeling approach and was intercepted for the NER crown by the larger model presented in DaCy. Some parts of the pretraining approach also seemed misguided on the small dataset. To further understand the DaLUKE performance, results from other downstream tasks than NER would be of great interest, including coreference resolution on the Dacoref dataset [KL04; Bro+21] or dependency parsing on the Danish Dependency Treebank [Kro03]. It would also improve insight to pretrain DaLUKE with a subset of the corpus left out as a testing set. Some further changes which have the potential to improve performance are proposed.

Data As in any deep learning project, we end up crying out for more data. Multiple of the pretraining experiments presented in Section 6.1 show signs of the model overfitting to the limited Danish Wikipedia. An immediate idea is to include the Norweigan Wikipedia as the primary language, *bokmål*, is syntactically, and vocabulary-wise similar to Danish. There are some technical issues such as the choice of tokenizer and entity modeling, as the same entity might be spelled differently in both languages. While the Norwegian Wikipedia has 156M words[6.6]compared to the 81M of the Danish Wikipedia, this might be too little to dramatically change the results.

A more impactful improvement could be found from our dataset augmentation idea, though it did not improve results for the Danish Wikipedia. Much needed entirely new LUKE pretraining datasets could be produced from raw text corpora by automatically annotating them using pattern matching. Such lower standard annotations used to achieve extra data might be necessary for continued improvement in low-resource languages such as Danish but should be used carefully and with an analysis of the trade-off between quality and quantity of data. Other automatic annotators than simple pattern matching might also be beneficial as the field of entity linking has been applied to Wikipedia [BB21] including ready-to-use tools such as the Danish version of DBpedia Spotlight [Dai+13]. It might hold that the quality of the entity annotations is not all-important, in which case tools that link entire sentences or paragraphs to entities without localizing them exactly might be used. The entity position IDs could then be omitted. Here, Danish tools for explicit semantic analysis [NH17] could help.

The primary, open dataset in which we see promise for automated annotation is the recently published Danish Gigaword corpus which includes documents that are more diverse in domain and dialect than the Danish Wikipedia [Der+21].

Access to closed datasets such as news articles and internal documents could also be a way to increase the available data. However, such documents are closed for a reason. In most cases, suppliers of closed documents would have an interest in not risking that these documents could be reverse engineered from the model. For this, methods such as differential privacy [Gon+20] could play a role.

Finally, investigating biases in the pretraining dataset could provide insight into why performance is better on some categories than others (see Table 5.3). Obviously, Wikipedia does not categorize articles by the four CoNLL-2003 categories, but many articles are still categorized into larger groups that could light a path to putting all or most articles into the CoNLL boxes.

[6.6]Norwegian Wikipedia statistics: `https://no.wikipedia.org/wiki/Spesial:Statistikk`

Modeling Motivated by the impressive success of DaCy, the most direct model improvement is to use the multilingual RoBERTa [Con+20] instead of the Danish BERT [Bot19]. The Danish BERT has previously been criticized for being under-trained [Der+21; NV20] and for not being case-sensitive. Furthermore, there exists a large version of the multilingual RoBERTa, and, finally, we speculate that our pretraining task will help the multilingual model catch up to possibly missing Danish-specific knowledge. The large version will, however, require more computational resources or more patience.

Further improvement might be found if the LUKE approach is not followed directly. The LUKE knowledge-enhancement is very subtle and requires any hierarchical and relational modeling of knowledge to be learned implicitly from the flat world of entities stored in the vocabulary. A unification of models that use subtle knowledge-enhancement in pretraining and those that use explicit knowledge bases might be beneficial for Danish. An approach to this that maintains a flexible, general model, might be introducing additional tokens coding for knowledge as input for the entity embeddings. We imagine a hierarchical token that uses the article classification of Wikipedia (see Figure 6.9) to code the type of token to be beneficial in pretraining.

Figure 6.9: An example of categories included in Wikipedia articles. It is imagined that these can be mined as additional, helpful annotations of entity types for the pretraining.

Ethics As with many applications of AI, there are significant ethical concerns. High-performing language models capable of generating convincing text can be used for automating the production of spam and other deceitful internet content. For instance, OpenAI claimed that GPT-2 [Rad+19] generated problematic articles so easily that its release was considerably delayed[6.7]. Incorporating knowledge in language models might allow bad actors better control over this production, requiring monitoring of the potentially harmful consequences of effective NLP.

[6.7] `https://www.technologyreview.com/2019/08/29/133218/openai-released-its-fake-news-ai-gpt-2/`. Visited June 26, 20201.

Another concern is that of unintended offensive or politically charged language. To acquire the amounts of data necessary to train effective language models, many practitioners, including BotXO for da-BERT [Bot19], have turned to uncurated web scrapes. As the internet has no shortage of hate, the models will naturally learn these parts of language. Such behaviour has for instance been observed with GPT-3 [Bro+20], the largest language model in existence as of writing[6.8]. One solution could be the use of NLP methods such as hate speech detection, as done for Danish by Sigurbergsson and Derczynski [SD20]. However, these problems of language models can also be seen in a larger AI safety context as an alignment problem [Tay+20]: How do we guarantee that the implicit values of these automated systems are compatible with our engineering goals of benefiting society?

[6.8] `https://www.technologyreview.com/2020/10/23/1011116/chatbot-gpt3-openai-facebook-google-safety-fix-racist-sexist-language-ai/`. Visited June 26, 2021.

7. Conclusion

We present DaLUKE, a Danish version of the LUKE [Yam+20], a general-purpose language model for producing contextualized word and entity representations. It is fine-tuned on three Danish named entity recognition (NER) datasets and achieves competitive performance on DaNE, the primary Danish NER dataset. Many Danish NER results are reproduced, revealing a group of close top contenders that includes DaLUKE. On raw scores, DaLUKE is superseded by DaCy large [Ene21], but beats DaNLP's fine-tuned version of BotXO's Danish BERT [Bro+21; Bot19], suggesting that the knowledge-based additions of Yamada et al. [Yam+20] do indeed raise the performance of the model. Furthermore, both released LUKE models (large and base) are fine-tuned on the CoNLL-2003 NER dataset [TD03], and the results reported by Yamada et al. are reproduced.

Both the general-purpose pretrained and the fine-tuned DaLUKE are made publicly available under the open source MIT license along with a `pip` installable package, `daluke`, that allows for easy use and integration into existing code bases.

Using DaNE as a benchmark, it was found that entity-aware self-attention and a complete entity vocabulary help learning, and that especially transfer learning is important to the performance of the model. Without transfer learning, the model seems to overfit to the pretraining dataset, which is a concern with small datasets in low-resource languages like Danish. Our data-engineering, however, gave an example of how changes in pretraining datasets may have unintuitive consequences and should be applied carefully.

The NER fine-tuning, used as a benchmark for language understanding, was found to be sensitive to hyperparameters and random number generator seeds, requiring broader analysis of Danish language models. However, it was found that the representation geometry of DaLUKE exhibits semantic language structure and that the model successfully can include context and knowledge in language predictions.

All in all, the knowledge augmenting methods of DaLUKE show positive results with several paths worth exploring to further empower low resource natural language processing.

Bibliography

[Akb+19] Alan Akbik et al. "FLAIR: An easy-to-use framework for state-of-the-art NLP". In: *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 2019, pp. 54–59.

[Al-+15] Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. "Polyglot-NER: Massive Multilingual Named Entity Recognition". In: *Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, British Columbia, Canada, April 30 - May 2, 2015* (Apr. 2015).

[Bat18] David S. Batista. "Named-Entity evaluation metrics based on entity-level". In: *David S. Batista Blog* (May 9, 2018). URL: `http://www.davidsbatista.net/blog/2018/05/09/Named_Entity_Evaluation/` (visited on 06/19/2021).

[BB21] Robin Brochier and Frédéric Béchet. "Predicting Links on Wikipedia with Anchor Text Information". In: *CoRR* abs/2105.11734 (2021). arXiv: `2105.11734`. URL: `https://arxiv.org/abs/2105.11734`.

[Bir20] Victor Elkjær Birk. "Investigating state-of-the-art approaches to knowledge enhancing deep learning based language models". MA thesis. Kongens Lyngby: Technical University of Denmark, 2020.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[Bot19] BotXO. *BotXO has trained the most advanced Danish BERT model to date*. 2019. URL: `https://www.botxo.ai/en/blog/danish-bert-model/` (visited on 02/28/2021).

[Bro+18] Per B. Brockhoff et al. "Introduction to Statistics at DTU". In: (2018). URL: `https://02323.compute.dtu.dk/enotes/book-IntroStatistics`.

[Bro+20] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: (2020). arXiv: `2005.14165 [cs.CL]`.

[Bro+21] Amalie Brogaard Pauli, Maria Barrett, Ophélie Lacroix, and Rasmus Hvingelby. "DaNLP: An open-source toolkit for Danish Natural Language Processing". In: *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa 2021)*. 2021.

[Cona] PyTorch Contributors. *Automatic Mixed Precision Package - torch.cuda.amp*. URL: `https://pytorch.org/docs/stable/amp.html` (visited on 05/27/2021).

[Conb] PyTorch Contributors. *CrossEntropyLoss*. URL: `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html` (visited on 06/09/2021).

[Conc] PyTorch Contributors. *Reproducability*. URL: https://pytorch.org/docs/stable/notes/randomness.html (visited on 06/09/2021).

[Con+20] Alexis Conneau et al. *Unsupervised Cross-lingual Representation Learning at Scale*. 2020. arXiv: 1911.02116 [cs.CL].

[Con21] Papers With Code Contributors. *Papers With Code: Named Entity Recognition*. 2021. URL: https://paperswithcode.com/task/named-entity-recognition-ner (visited on 06/02/2021).

[Dai+13] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. "Improving Efficiency and Accuracy in Multilingual Entity Extraction". In: *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*. 2013.

[Der+21] Leon Derczynski et al. "The Danish Gigaword Corpus". In: *Proceedings of the 23rd Nordic Conference on Computational Linguistics*. NEALT, 2021.

[Dev+19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

[DFB14] Leon Derczynski, Camilla Vilhelmsen Field, and Kenneth S. Bøgh. "DKIE: Open Source Information Extraction for Danish". In: *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Gothenburg, Sweden: Association for Computational Linguistics, Apr. 2014, pp. 61–64. DOI: 10.3115/v1/E14-2016. URL: https://www.aclweb.org/anthology/E14-2016.

[Ene21] Kenneth Enevoldsen. "DaCy: A SpaCy NLP Pipeline for Danish". In: 2021.

[Fou04] The Apache Software Foundation. *Apache License, Version 2.0*. 2004. URL: https://www.apache.org/licenses/LICENSE-2.0.html (visited on 06/11/2021).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[Gon+20] Maoguo Gong et al. "Preserving differential privacy in deep neural networks with relevance-based adaptive noise imposition". In: *Neural Networks* 125 (2020), pp. 131–141. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2020.02.001. URL: https://www.sciencedirect.com/science/article/pii/S0893608020300460.

[Gra+18] Edouard Grave et al. "Learning Word Vectors for 157 Languages". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. URL: https://www.aclweb.org/anthology/L18-1550.

[He+20] Bin He, Xin Jiang, Jinghui Xiao, and Qun Liu. "KgPLM: Knowledge-guided Language Model Pre-training via Generative and Discriminative Learning". In: *CoRR* abs/2012.03551 (2020). arXiv: 2012.03551. URL: https://arxiv.org/abs/2012.03551.

[Hed+21] Michael A. Hedderich et al. *A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios*. 2021. arXiv: 2010.12309 [cs.CL].

[HG20] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. 2020. arXiv: 1606.08415 [cs.LG].

[Høj20] Malte Højmark-Bertelsen. "Ælæctra - A Step Towards More Efficient Danish Natural Language Processing". Bachelor's Thesis. Cognitive Science, Aarhus University, 2020.

[Hon+20] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. *spaCy: Industrial-strength Natural Language Processing in Python*. 2020. DOI: 10.5281/zenodo.1212303. URL: https://doi.org/10.5281/zenodo.1212303.

[HR18] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 328–339. DOI: 10.18653/v1/P18-1031. URL: https://www.aclweb.org/anthology/P18-1031.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation*. MIT Press, 1997, pp. 1735–1780. URL: https://direct.mit.edu/neco/article/9/8/1735/6109/Long-Short-Term-Memory (visited on 06/18/2021).

[HTC20] Mengdi Huang, Chetan Tekur, and Michael Carilli. *Introducing native PyTorch automatic mixed precision for faster training on NVIDIA GPUs*. July 2020. URL: https://pytorch.org/blog/accelerating-training-on-nvidia-gpus-with-pytorch-automatic-mixed-precision/ (visited on 05/28/2021).

[Hvi+20] Rasmus Hvingelby et al. "DaNE: A Named Entity Resource for Danish". English. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 4597–4604. ISBN: 979-10-95546-34-4. URL: https://www.aclweb.org/anthology/2020.lrec-1.565.

[JMP15] Anders Johannsen, Hector Martinez, and Alonso Barbara Plank. "Universal Dependencies for Danish". In: *TLT14*. 2015.

[KH91] Anders Krogh and John A. Hertz. "A Simple Weight Decay Can Improve Generalization". In: *Proceedings of the 4th International Conference on Neural Information Processing Systems*. NIPS'91. Denver, Colorado: Morgan Kaufmann Publishers Inc., 1991, pp. 950–957. ISBN: 1558602224.

[Kir+19] Andreas Kirkedal, Barbara Plank, Leon Derczynski, and Natalie Schluter. "The Lacunae of Danish Natural Language Processing". In: *Proceedings of the 22nd Nordic Conference on Computational Linguistics*. Turku, Finland: Linköping University Electronic Press, Sept. 2019, pp. 356–362. URL: https://www.aclweb.org/anthology/W19-6141.

[KL04] M.T. Kromann and S.K. Lynge. *Copenhagen Dependency Treebank*. 2004. URL: https://github.com/mbkromann/copenhagen-dependency-treebank (visited on 06/24/2021).

[Kle11] Joel F. Klein. *Amdahl's Law*. Mar. 2011. URL: https://demonstrations.wolfram.com/AmdahlsLaw/ (visited on 06/11/2021).

[KN20] Lars Kjeldgaard and Lukas Nielsen. "NERDA". In: GitHub, 2020. URL: https://github.com/ebanalyse/NERDA.

[Kro03] Matthias Trautner Kromann. "The Danish Dependency Treebank and the DTAG treebank tool". In: *In Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003*. 2003, pp. 14–15.

[Lau17] Jon Launchbury. *DARPA Perspective on AI*. 2017. URL: https://www.darpa.mil/about-us/darpa-perspective-on-ai (visited on 05/20/2021).

[LBH15] Yann LeCun, Y. Bengio, and Geoffrey Hinton. "Deep Learning". In: *Nature* 521 (May 2015), pp. 436–44. DOI: 10.1038/nature14539.

[LH19] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

[Liu+19] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: 2019. arXiv: 1907.11692 [cs.CL].

[Log+19] Robert Logan et al. "Barack's Wife Hillary: Using Knowledge Graphs for Fact-Aware Language Modeling". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 5962–5971. DOI: 10.18653/v1/P19-1598. URL: https://www.aclweb.org/anthology/P19-1598.

[Man+14] Christopher D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, pp. 55–60. URL: http://www.aclweb.org/anthology/P/P14/P14-5010.

[Mar+13] Mónica Marrero et al. "Named Entity Recognition: Fallacies, challenges and opportunities". In: *Computer Standards and Interfaces* 35.5 (2013), pp. 482–489. ISSN: 0920-5489. DOI: https://doi.org/10.1016/j.csi.2012.09.004. URL: https://www.sciencedirect.com/science/article/pii/S0920548912001080.

[McI+18] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. "UMAP: Uniform Manifold Approximation and Projection". In: *The Journal of Open Source Software* 3.29 (2018), p. 861.

[MH08] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605. URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[MHM20] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML].

[MIT] MIT. *The MIT License (MIT)*. URL: https://mit-license.org/ (visited on 05/29/2021).

[Nak18] Hiroki Nakayama. *seqeval: A Python framework for sequence labeling evaluation*. Software available from https://github.com/chakki-works/seqeval. 2018. URL: https://github.com/chakki-works/seqeval.

[NH17] Finn Årup Nielsen and Lars Kai Hansen. "Open semantic analysis: The case of word level semantics in Danish". English. In: *Proceedings of 8th Language and Technology Conference*. 8th Language and Technology Conference , LTC 2017 ; Conference date: 17-11-2017 Through 19-11-2017. 2017.

[Nie21] Finn Årup Nielsen. *Awesome Danish: A curated list of awesome resources for Danish language technology*. 2021. URL: https://github.com/fnielsen/awesome-danish/blob/master/README.md (visited on 06/11/2021).

[NKA98] Ole Norling-Christensen, Britt-Katrin Keson, and Jørg Asmussen. *PAROLE-DK and ePAROLE: Morphosyntactically Annotated Danish Language Corpus*. 1998.

[NV20] Lukas Christian Nielsen and Sebastian Lindegaard Veile. "Automatic Text Summarization For Danish Using BERT". Master's Thesis. IT University of Copenhagen, 2020.

[OMK18] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. "A Survey of the Usages of Deep Learning in Natural Language Processing". In: *CoRR* abs/1807.10854 (2018). arXiv: 1807.10854. URL: http://arxiv.org/abs/1807.10854.

[Pan+17] Xiaoman Pan et al. "Cross-lingual Name Tagging and Linking for 282 Languages". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1946–1958. DOI: 10.18653/v1/P17-1178. URL: https://www.aclweb.org/anthology/P17-1178.

[Ped+11] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[Pet+19] Matthew E. Peters et al. "Knowledge Enhanced Contextual Word Representations". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 43–54. DOI: 10.18653/v1/D19-1005. URL: https://www.aclweb.org/anthology/D19-1005.

[Pla19] Barbara Plank. "Neural Cross-Lingual Transfer and Limited Annotated Data for Named Entity Recognition in Danish". In: *Proceedings of the 22nd Nordic Conference on Computational Linguistics*. Turku, Finland: Linköping University Electronic Press, Sept. 2019, pp. 370–375. URL: https://www.aclweb.org/anthology/W19-6143.

[Rad+18] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding with unsupervised learning". In: (2018). URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (visited on 06/24/2021).

[Rad+19] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: 2019.

[Raj19] Bharat S Raj. *Understanding BERT: Is it a Game Changer in NLP?* 2019. URL: https://towardsdatascience.com/understanding-bert-is-it-a-game-changer-in-nlp-7cca943cf3ad (visited on 06/24/2021).

[RLC19] Afshin Rahimi, Yuan Li, and Trevor Cohn. "Massively Multilingual Transfer for NER". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 151–164. URL: https://www.aclweb.org/anthology/P19-1015.

[RM95] Lance A. Ramshaw and Mitchell P. Marcus. "Text Chunking using Transformation-Based Learning". In: *CoRR* cmp-lg/9505040 (1995). URL: http://arxiv.org/abs/cmp-lg/9505040.

[RN09] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: a modern approach*. 3rd ed. Pearson, 2009.

[Ros+20] Corby Rosset et al. "Knowledge-Aware Language Model Pretraining". In: *CoRR* abs/2007.00655 (2020). arXiv: 2007.00655. URL: https://arxiv.org/abs/2007.00655.

[Rud21] Sebastian Ruder. *NLP-progress: Named Enitty Recognition*. 2021. URL: http://nlpprogress.com/english/named_entity_recognition.html (visited on 06/02/2021).

[SD20] Gudbjartur Ingi Sigurbergsson and Leon Derczynski. "Offensive Language and Hate Speech Detection for Danish". English. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 3498–3508. ISBN: 979-10-95546-34-4. URL: https://www.aclweb.org/anthology/2020.lrec-1.430.

[SHB16] Rico Sennrich, Barry Haddow, and Alexandra Birch. "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)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://www.aclweb.org/anthology/P16-1162.

[SM18] Mohammad Golam Sohrab and Makoto Miwa. "Deep Exhaustive Model for Nested Named Entity Recognition". In: *Proceedings of the 2018 Confer- ence on Empirical Methods in Natural Language Processing*. 2018, pp. 2843–2849. URL: https://www.aclweb.org/anthology/D18-1309.pdf (visited on 05/30/2021).

[SP97] M. Schuster and K.K. Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681. DOI: 10.1109/78.650093.

[Tay+20] Jessica Taylor, Eliezer Yudkowsky, Patrick LaVictoire, and Andrew Critch. "Alignment for Advanced Machine Learning Systems". In: 2020.

[TD03] Erik F. Tjong Kim Sang and Fien De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 2003, pp. 142–147. URL: https://www.aclweb.org/anthology/W03-0419.

[Vas+17] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[Wan+18] Alex Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. DOI: 10.18653/v1/W18-5446. URL: https://www.aclweb.org/anthology/W18-5446.

[Wan+20] Ruize Wang et al. "K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters". In: *CoRR* abs/2002.01808 (2020). arXiv: 2002.01808. URL: https://arxiv.org/abs/2002.01808.

[Wan+21] Xiaozhi Wang et al. "KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation". In: *Transactions of the Association for Computational Linguistics* 9 (Mar. 2021), pp. 176–194. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00360. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00360/1894315/tacl_a_00360.pdf. URL: https://doi.org/10.1162/tacl%5C_a%5C_00360.

[Wik21] Wikipedia Contributors. *Named-entity recognition — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Named-entity_recognition&oldid=1020618016. [Online; accessed 1-June-2021]. 2021.

[Wu+16] Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144 (2016). URL: http://arxiv.org/abs/1609.08144.

[Xio+19] Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. *Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model*. 2019. arXiv: 1912.09637 [cs.CL].

[Yam+20] Ikuya Yamada et al. "LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 6442–6454. DOI: 10.18653/v1/2020.emnlp-main.523. URL: https://www.aclweb.org/anthology/2020.emnlp-main.523.

[Zha+19] Zhengyan Zhang et al. "ERNIE: Enhanced Language Representation with Informative Entities". In: *Proceedings of ACL 2019*. 2019.

A. Result Details

A.1 Experiment: No Weight Fixing in Pretraining

This experiment was conducted as an ablation study on the locking of da-BERT weights. All hyperparameters are the same as in the baseline experiment, described in Section 6.1.3, with the exception that the weights from da-BERT were unlocked for the entire pretraining.

Model	Top k accuracy [%]	$k = 1$	$k = 3$	$k = 5$	$k = 10$	$k = 25$	$k = 50$
Baseline	Masked words	24.4	31.2	34.6	39.9	47.7	53.7
	Masked entities	37.7	48.7	53.6	60.3	68.7	75.0

Table A.1: The top k accuracy of the model with no parameter fixing in the 50'th and last epoch.

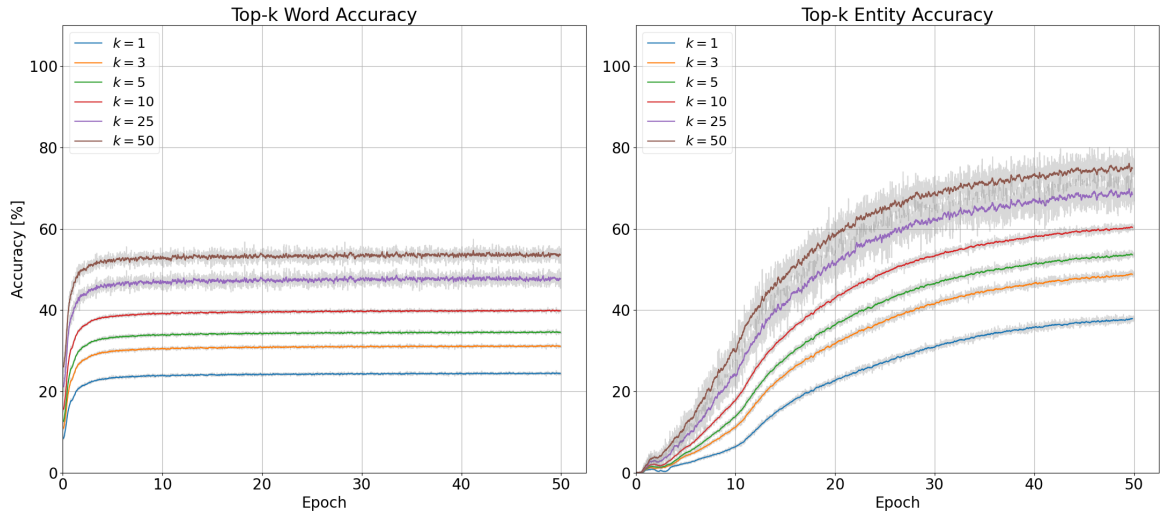

Figure A.1: Development of masked language and masked entities when all weights are unlocked for the entire pretraining.

Model	F1 [%]					Precision [%]	Recall [%]
	Avg.	LOC	PER	ORG	MISC	Avg.	Avg.
No weight fixing	81.8	87.9	93.8	73.3	68.7	85.2	78.7

Table A.2: The fine-tuning results of the weight unlocking pretraining experiment.

A.2 Fine-tuning Hyperparameter Search Results

Batch size	Learning rate	Weight decay	Dropout	Loss weight	F1 [%]	Precision	Recall
8	$1{\cdot}10^{-5}$	0.01	0.025	Yes	73.06	62.82	87.29
8	$1{\cdot}10^{-5}$	0.01	0.025	No	73.53	62.87	88.54
8	$1{\cdot}10^{-5}$	0.01	0.1	Yes	73.52	63.17	87.92
8	$1{\cdot}10^{-5}$	0.01	0.1	No	71.39	60.00	88.12
8	$1{\cdot}10^{-5}$	0.05	0.025	Yes	72.87	62.54	87.29
8	$1{\cdot}10^{-5}$	0.05	0.025	No	67.77	56.07	85.62
8	$1{\cdot}10^{-5}$	0.05	0.1	Yes	71.44	60.46	87.29
8	$1{\cdot}10^{-5}$	0.05	0.1	No	71.91	61.45	86.67
8	$5{\cdot}10^{-5}$	0.01	0.025	Yes	83.08	77.14	90.00
8	$5{\cdot}10^{-5}$	0.01	0.025	No	**85.35**	**80.82**	90.42
8	$5{\cdot}10^{-5}$	0.01	0.1	Yes	84.96	80.45	90.00
8	$5{\cdot}10^{-5}$	0.01	0.1	No	31.53	19.86	76.46
8	$5{\cdot}10^{-5}$	0.05	0.025	Yes	84.00	79.41	89.17
8	$5{\cdot}10^{-5}$	0.05	0.025	No	84.24	77.78	**91.88**
8	$5{\cdot}10^{-5}$	0.05	0.1	Yes	84.65	79.74	90.21
8	$5{\cdot}10^{-5}$	0.05	0.1	No	**85.35**	**80.82**	90.42
16	$1{\cdot}10^{-5}$	0.01	0.025	Yes	65.29	52.76	85.62
16	$1{\cdot}10^{-5}$	0.01	0.025	No	63.91	51.12	85.21
16	$1{\cdot}10^{-5}$	0.01	0.1	Yes	63.33	50.24	85.62
16	$1{\cdot}10^{-5}$	0.01	0.1	No	62.57	49.57	84.79
16	$1{\cdot}10^{-5}$	0.05	0.025	Yes	63.73	50.61	86.04
16	$1{\cdot}10^{-5}$	0.05	0.025	No	63.63	50.55	85.83
16	$1{\cdot}10^{-5}$	0.05	0.1	Yes	64.13	51.11	86.04
16	$1{\cdot}10^{-5}$	0.05	0.1	No	63.06	49.70	86.25
16	$5{\cdot}10^{-5}$	0.01	0.025	Yes	84.83	79.10	91.46
16	$5{\cdot}10^{-5}$	0.01	0.025	No	83.72	78.26	90.00
16	$5{\cdot}10^{-5}$	0.01	0.1	Yes	82.66	76.88	89.38
16	$5{\cdot}10^{-5}$	0.01	0.1	No	29.21	18.08	76.04
16	$5{\cdot}10^{-5}$	0.05	0.025	Yes	83.67	77.25	91.25
16	$5{\cdot}10^{-5}$	0.05	0.025	No	83.60	76.70	**91.88**
16	$5{\cdot}10^{-5}$	0.05	0.1	Yes	83.29	76.90	90.83
16	$5{\cdot}10^{-5}$	0.05	0.1	No	82.94	76.45	90.62

Table A.3: The results of hyperparameter search with micro average percentages for the metrics reported on the DaNE test set.

B. Additional Figures

B.1 Dimensionality Reduction

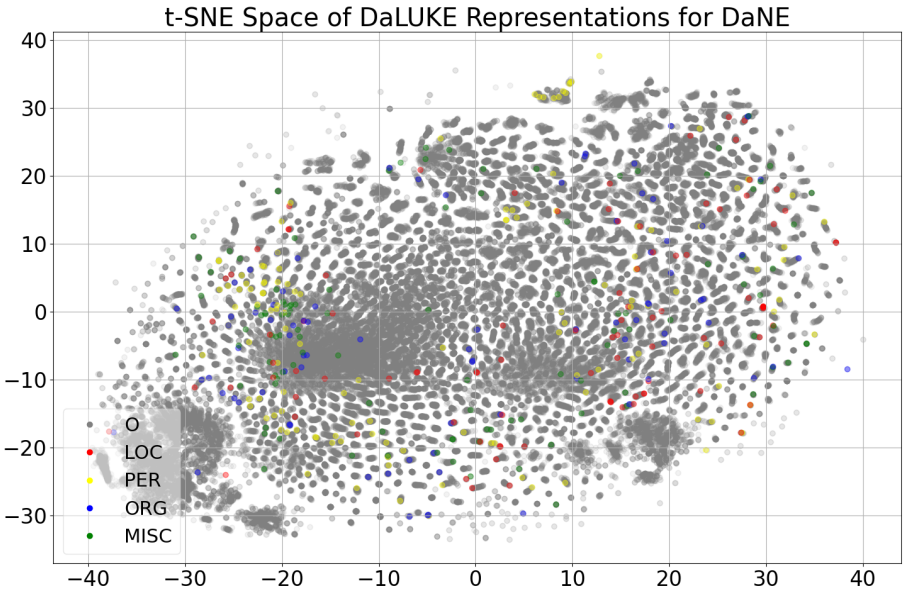

Figure B.1: t-SNE performed on a 100K random subset of the full$\sim$ 1M possible entity spans in the training set of DaNE.

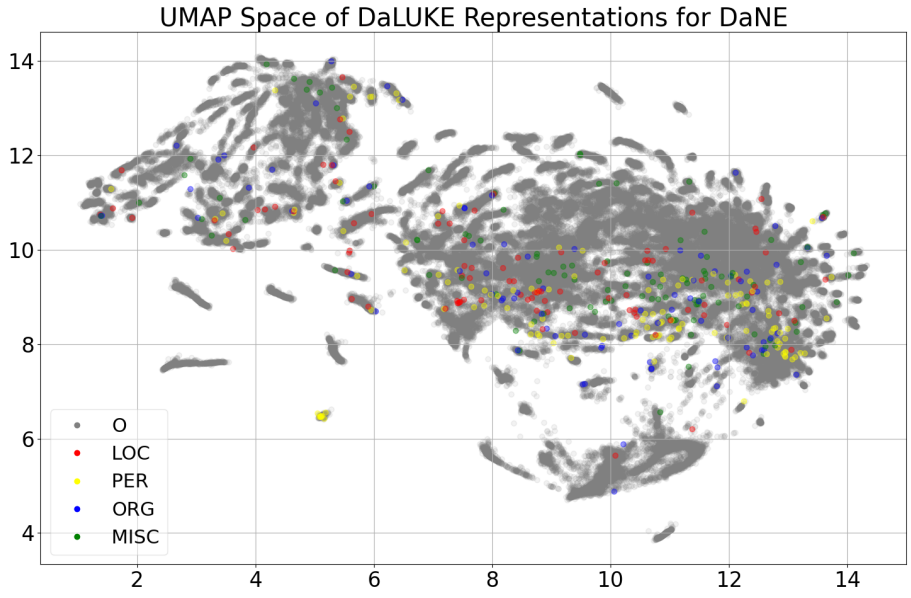

Figure B.2: UMAP performed on a 100K random subset of the full $\sim$ 1M possible entity spans in the DaNE training set.

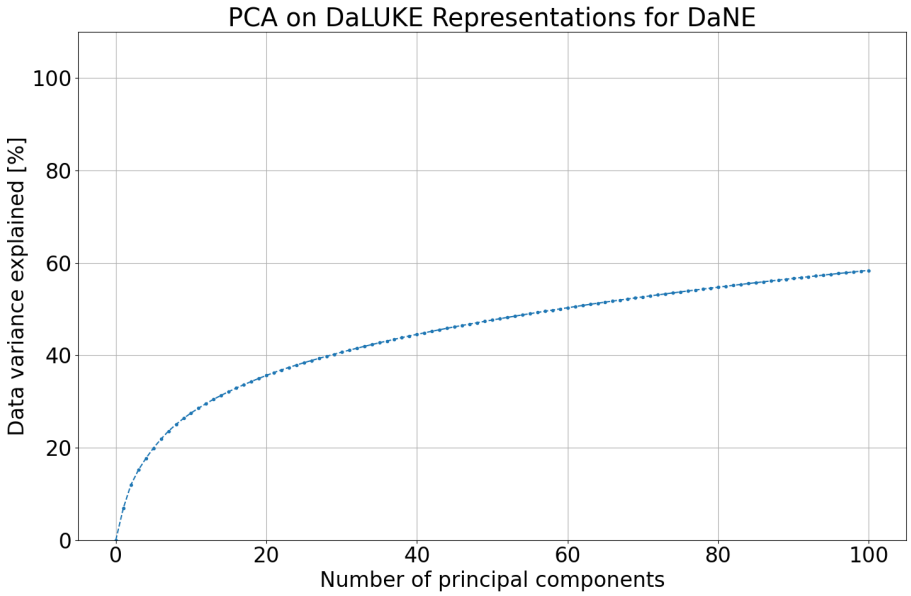

Figure B.3: Variance explained by principal components found by PCA on full DaNE training dataset.

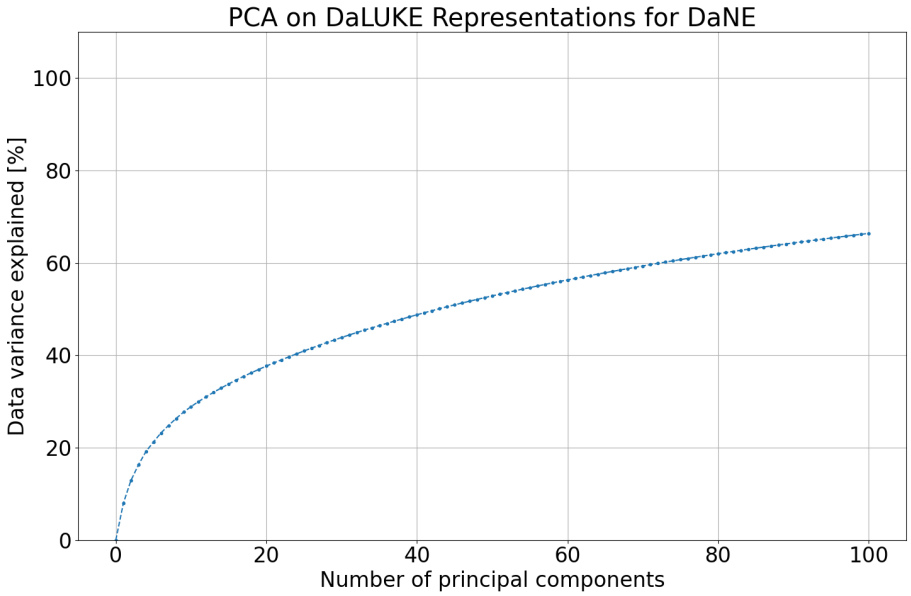

Figure B.4: Variance explained by principal components found by PCA on the dataset including positive labels.