information

Article

Optimizing Small BERTs Trained for German NER

Jochen Zöllner [1,2,*], Konrad Sperfeld [1], Christoph Wick [2] and Roger Labahn [1]

[1] Institute of Mathematics, University of Rostock, 18057 Rostock, Germany; konrad.sperfeld@uni-rostock.de (K.S.); roger.labahn@uni-rostock.de (R.L.)
[2] PLANET AI GmbH Rostock, 18057 Rostock, Germany; christoph.wick@planet-ai.de
* Correspondence: jochen.zoellner@uni-rostock.de

Abstract: Currently, the most widespread neural network architecture for training language models is the so-called BERT, which led to improvements in various Natural Language Processing (NLP) tasks. In general, the larger the number of parameters in a BERT model, the better the results obtained in these NLP tasks. Unfortunately, the memory consumption and the training duration drastically increases with the size of these models. In this article, we investigate various training techniques of smaller BERT models: We combine different methods from other BERT variants, such as ALBERT, RoBERTa, and relative positional encoding. In addition, we propose two new fine-tuning modifications leading to better performance: Class-Start-End tagging and a modified form of Linear Chain Conditional Random Fields. Furthermore, we introduce Whole-Word Attention, which reduces BERTs memory usage and leads to a small increase in performance compared to classical Multi-Head-Attention. We evaluate these techniques on five public German Named Entity Recognition (NER) tasks, of which two are introduced by this article.

Keywords: named entity recognition; natural language processing; BERT; German language; pre-training; fine-tuning; dataset

check for updates

Citation: Zöllner, J.; Sperfeld, K.; Wick, C.; Labahn, R. Optimizing Small BERTs Trained for German NER. *Information* **2021**, *12*, 443. https://doi.org/10.3390/info12110443

Academic Editor: Diego Reforgiato Recupero

Received: 28 September 2021
Accepted: 20 October 2021
Published: 25 October 2021

1. Introduction

NER is a well-known task in the field of NLP. The NEISS project [1] in which we work in close cooperation with Germanists is devoted to the automation of diverse processes during the creation of digital editions. One key task in this area is the automatic detection of entities in text corpora that correspond to a common NER task. Currently, the best results for NER tasks have been achieved with Transformer-based [2] language models, such as Bidirectional Encoder Representations from Transformers (BERT) [3]. Classically, a BERT is first pre-trained with large amounts of unlabeled text to obtain a robust language model and then fine-tuned to a downstream task. In particular, for the pre-training step, many variants of BERT, such as ALBERT [4], RoBERTa [5], or XLNet [6], were already investigated. Pre-training is resource-intensive and takes a long time (several weeks) for training. For that reason, online platforms, such as Hugging Face [7], offer a zoo of already pre-trained networks that can be directly used to train a downstream task. However, the available models are not always suitable for a certain task, such as NER, in German because they can be pre-trained on a different domain (e.g., language, time epoch, or text style).

Furthermore, when philologists create new digital editions, different research priorities can be set so that a different associated NER task is created each time. That is why philologists must also be able to train individual NER tasks themselves who commonly only have access to limited compute resources. For this reason, the aim is to train NER tasks on smaller BERT models as best as possible. Since our focus is the NER, we test if the optimizations work consistently on five different German NER tasks. Due to our project aims, we evaluated our new methods on the German language. We suspect a consistent behavior on similar European languages such as English, French and Spanish. Two of the considered tasks rely on new NER datasets that we generated from existing digital text editions.

Therefore, in this article, we examine which techniques are optimal to pre-train and fine-tune a BERT to solve NER tasks in German with limited resources. We investigate this on smaller BERT models with six layers that can be pre-trained on a single GPU (RTX 2080 Ti 11 GB) within 9 days, whereas fine-tuning can be performed on a notebook CPU in a few hours.

We first compared different well-established pre-training techniques, such as Mask Language Model (MLM), Sentence Order Prediction (SOP), and Next Sentence Prediction (NSP), on the final result of the downstream NER task. Furthermore, we investigated the influence of absolute and relative positional encoding, as well as Whole-Word Masking (WWM).

As a second step, we compared various approaches for carrying out fine-tuning since the tagging rules cannot be learned consistently by classical fine-tuning approaches. In addition to existing approaches, such as the use of Linear Chain Conditional Random Fields (LCRFs), we propose the so-called Class-Start-End (CSE) tagging and a specially modified form of LCRFs for NER, which led to an increased performance. Furthermore, for decoding, we introduced a simple rule-based approach, which we call the Entity-Fix rule, to further improve the results.

As already mentioned, the training of a BERT requires many resources. One of the reasons is that the memory amount of BERT depends quadratically on the sequence length when calculating the energy values (attention scores) in its attention layers, which leads to memory problems for long sequences. In this article, we propose Whole-Word Attention, a new modification of the Transformer architecture that not only reduces the number of energy values to be calculated by about a factor of two but also results in slightly improved results.

In summary, the main goal of this article is to enable the training of efficient BERT models for German NER on limited resources. For this, the article provides different methodology and claims the following contributions:

- We introduce and share two datasets for German NER formed from existing digital editions.
- We investigate the influence of different BERT pre-training methods, such as pre-training tasks, varying positional encoding, and adding Whole-Word Masking on a total of five different NER datasets.
- On the same NER tasks, we investigate different approaches to perform fine-tuning. Hereby, we propose two new methods that led to performance improvements: Class-Start-End tagging and a modified form of Linear Chain Conditional Random Fields.
- We introduce a novel rule-based decoding strategy achieving further improvements.
- We propose Whole-Word Attention, a modification of the BERT architecture that reduces the memory requirements of the BERT models, especially for processing long sequences, and also leads to further performance improvements.
- We share the datasets (see Section 2) and our source code (https://github.com/NEISSproject/tf2_neiss_nlp/tree/berNer21, accessed on 19 October 2021), which is based on tfaip [8] with the community.

The remainder of this article is structured as follows: In Section 2, we present our datasets, including the two new German NER datasets. In Section 3, we introduce the different pre-training techniques, and Section 4 describes fine-tuning. Subsequently, in Section 5, we introduce Whole-Word Attention (WWA). In all these sections, we provide an overview of the existing techniques with the corresponding related work that we adopted and also introduce our novel methods. Afterwards, Section 6 shows the conducted experiments and their results. We conclude this article with a discussion of our results and providing an outlook on future work.

2. Datasets

In this section, we list the different datasets. First, we describe the dataset used for pre-training throughout our experiments. Then, we mention the key attributes of five NER datasets for the downstream tasks.

2.1. Pre-Training Data

To pre-train a BERT, a large amount of unlabeled text is necessary as input data. We collected the German Wikipedia and a web crawl of various German newspaper portals to pre-train our BERT. The dump of the German Wikipedia was preprocessed by the Wiki-Extractor [9], resulting in about 6 GB of text data. In addition, we took another 2 GB of German text data from different newspaper portals. We used various German newspaper portals, such as https://www.faz.net/aktuell/ (accessed on 19 October 2021) or https://www.berliner-zeitung.de/ (accessed on 19 October 2021) in August 2020, crawled with the news-please framework [10].

2.2. NER Downstream-Datasets

We evaluated our methods on five different NER tasks. In addition to three already existing German NER datasets—the frequently used GermEval 2014 dataset and two NER datasets on German legal texts—we introduce two NER tasks of two existing digital editions. In the following, we describe each of the five tasks.

2.2.1. GermEval 2014

One of the most widespread German NER datasets is GermEval 2014 [11], which comprises several News Corpora and Wikipedia. In total, it contains about 590,000 tokens with about 41,000 entities that are tagged into four main entity classes: "person", "organization", "location", and "other". Each main class can appear in a default, a partial, or a derived variant, resulting in 12 overall classes. In the GermEval task, entities can be tagged in two levels: outer and inner (nested entities). Since there are few inner annotations in the dataset, we restrict ourselves to evaluating the outer entities in our experiments as it is often the approach in other papers (e.g., [12–14]). This is called the outer chunk evaluation scheme, which is described in more detail by [14].

2.2.2. Legal Entity Recognition

The Legal Entity Recognition (LER) dataset [15] contains 2.15 million tokens with 54,000 manually annotated entities from German court decision documents of 2017 and 2018. The entities are divided into seven main classes and 19 subclasses, which we label by Coarse-Grained (CG) and Fine-Grained (FG), respectively. The FG task (LER FG) is more difficult than the CG task (LER CG) due to its larger number of possible classes.

2.2.3. Digital Edition: Essays from H. Arendt

We created an NER dataset based on the digital edition "Sechs Essays" by H. Arendt. It consists of 23 documents from the period 1932–1976, which were published online in [16] as TEI files [17]. In these documents, certain entities were manually tagged. Since some of the original NER tags comprised too few examples and some ambiguities (e.g., place and country), we joined several tags, as shown in Table 1.

Note that we removed any annotation of the class "ship" since only four instances were available in the dataset and no other similar class is available. We provide the resulting dataset online (see https://github.com/NEISSproject/NERDatasets/tree/main/Arendt, accessed on 19 October 2021) in a format similar to the CONLL-X format [18] and in a simple JSON format under a CC BY-NC-SA 3.0 DE license together with the training, development, and test partitions. Since not all entities are equally distributed over the 23 documents, the sentences of all documents are shuffled before splitting them into partitions.

Table 1. Distribution of NER entities in H. Arendt Edition. Column "Original attributes" lists which attributes from the original TEI files were combined into one "Entity" for the NER dataset. On average, an entity consists of 1.36 words.

Entity	# All	# Train	# Test	# Devel	Original Attributes
person	1702	1303	182	217	person, biblicalFigure, ficticiousPerson, deity, mythologicalFigure
place	1087	891	111	85	place, country
ethnicity	1093	867	115	111	ethnicity
organization	455	377	39	39	organization
event	57	49	6	2	event
language	20	14	4	2	language
unlabeled words	153,223	121,154	16,101	15,968	

2.2.4. Digital Edition: Sturm Edition

The second NER dataset consists of 174 letters of the years 1914–1922 from the Sturm Edition [19] available online in TEI format. It is much simpler than the dataset from the H. Arendt edition and contains only persons, places, and dates as tagged entities. From the original TEI files, we built an NER dataset with tags distributed, as shown in Table 2. Similarly to the H. Arendt dataset, the resulting dataset is available online (see https://github.com/NEISSproject/NERDatasets/tree/main/Sturm, accessed on 19 October 2021) in a format similar to the CONLL-X format and in a simple JSON format under a CC-BY 4.0 license together with the training, development, and test partitions. In contrast to the H. Arendt dataset, we split the 174 letters without shuffling the sentences across all documents.

Table 2. Distribution of NER entities in the Sturm Edition. On average, an entity consists of 1.12 words.

Entity	# All	# Train	# Test	# Devel
person	930	763	83	84
date	722	612	59	51
place	492	374	59	59
unlabeled words	33,809	27,047	3306	3456

3. Pre-Training Techniques

In this section, we provide an overview of several common pre-training techniques for a BERT that we examined in our experiments.

3.1. Pre-Training Tasks

In the original BERT [3], pre-training is performed by simultaneously minimizing the loss of the so-called Mask Language Model (MLM) and Next Sentence Prediction (NSP) tasks. The MLM task first tokenizes the text input with a subword tokenizer, then 15% of the tokens are chosen randomly. Hereby, 80% of these chosen tokens are replaced by a special mask token, 10% are replaced by a randomly chosen other token, and the remaining 10% keep the original correct token. Therefore, the goal of the MLM task is to find the original token for the 15% randomly chosen tokens, which is only possible by understanding the language and thus learning a robust language model.

Since BERT should also be able to learn the semantics of different sentences within a text, NSP was additionally included. When combining NSP with MLM, the input for pre-training are two masked sentences that are concatenated and separated by a special separator token. In 50% of the cases, two consecutive sentences from the same text document are used, whereas, in the other 50%, two random sentences from different documents are selected. The goal of the NSP task is to identify which of the two variants it is.

In the follow-up papers, RoBERTa [5] and XLNet [6] experiments showed that the NSP task often had no positive effect on the performance of the downstream tasks. Therefore, both papers recommended that the pre-training should solely be performed by the MLM task. In the ALBERT paper [4], this was investigated in more detail. They assumed that the ineffectiveness of the NSP task was only due to its simplicity, which is why they introduced Sentence Order Prediction (SOP) as a more challenging task that aims to learn relationships between sentences similar to the NSP task: BERT always receives two consecutive sentences, but in 50% of the cases, the order is wrong by flipping them. The SOP task is to learn the correct order of the sentences.

In this article, we examine the influences of the different pre-training tasks (MLM, NSP, and SOP) with the focus on improving the training of BERT for German NER tasks.

3.2. Absolute and Relative Positional Encoding

The original Transformer architecture [2] was exclusively based on attention mechanisms to process input sequences. Attention mechanisms allow every sequence element to learn relations to all other elements. By default, attention does not take into account information about the order of the elements in the sequence. However, since information about the order of the input sequence elements is mandatory in almost every NLP task, the original Transformer architecture introduced the so-called absolute positional encoding: a fixed position vector $p_j \in \mathbb{R}^{d_{model}}$ was added to each embedded input sequence element x_j at position $j \in \{1, \dots, n\}$ for an input sequence of length n; thus

$$x'_j = x_j + p_j.$$

In the original approach, the position vector p_j is built by computing sinusoids of different wavelengths in the following way:

$$\begin{aligned} p_{j,2k} &:= \sin\left(j/10{,}000^{2k/d_{model}}\right), \\ p_{j,2k+1} &:= \cos\left(j/10{,}000^{2k/d_{model}}\right) \end{aligned}$$

where $k \in \left\{1, \dots, \lfloor \frac{d_{model}}{2} \rfloor \right\}$. While the experiments in [2] showed great results, the disadvantage of absolute positional encoding is that the performance is significantly reduced in cases where the models are applied on sequences longer than those on which they were trained because the respective position vectors were not yet seen during training. Therefore, in [20], other variants for positional encoding were investigated and compared on translation tasks. The most promising approach was relative positional encoding [21]: a trainable distance information d^K_{j-i} is added in the attention layer when computing the energy $e_{i,j}$ of the ith sequence element to the jth one. Thus, if x_i and x_j are the ith and jth input elements of a sequence in an attention layer, instead of multiplying just the query vector $W^Q x_i$ with the key vector $W^K x_j$, one adds the trainable distance information d^K_{j-i} to the key vector resulting in

$$e_{i,j} := \frac{\left(W^Q x_i\right)^T \left(W^K x_j + d^K_{j-i}\right)}{\sqrt{d_k}} \tag{1}$$

where $W^Q_n, W^K_n \in \mathbb{R}^{d_{model} \times d_k}$. In addition, when multiplying the energy (after applying softmax) with the values, other trainable distance information d^V_{j-i} is added. Finally, the output y_i for the ith sequence element of a sequence of length n with relative positional encoding is computed by

$$y_i = \sum_{j=1}^{n} \alpha_{i,j} \left(W^V x_j + d^V_{j-i}\right) \tag{2}$$

where $\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{n} \exp(e_{i,k})}$ and $W^V \in \mathbb{R}^{d_{model} \times d_v}$. To train d^K_{j-i} and d^V_{j-i}, a hyperparameter τ (called the clipping distance), the trainable embeddings $r^K_{-\tau}, \ldots, r^K_{\tau} \in \mathbb{R}^{d_k}$, and $r^V_{-\tau}, \ldots, r^V_{\tau} \in \mathbb{R}^{d_v}$ are introduced. These embeddings are used to define the distance terms d^K_{j-i} and d^V_{j-i}, where distances longer than the clipping distance τ are represented by r_τ or $r_{-\tau}$; thus:

$$\begin{aligned} d^K_{j'-j} &= r^K_{\mathrm{clip}_\tau(j'-j)} && (3) \\ d^V_{j'-j} &= r^V_{\mathrm{clip}_\tau(j'-j)} && (4) \\ \mathrm{clip}_\tau(x) &= \max(-\tau, \min(\tau, x)) \end{aligned}$$

The authors in [20] already showed that relative positional encoding suffers less from the disadvantages of absolute position encoding of unseen sequence lengths. In this article, we examine the influence of these two variants of positional encoding during the training of German BERT models.

3.3. Whole-Word Masking (WWM)

Whole-Word Masking (WWM) is a small modification of the Mask Language Model (MLM) task described in Section 3.1. In contrast to the classic MLM task, WWM does not mask token-wisely but instead word-wisely. This means that in all cases, either all tokens belonging to a word are masked or none of them. Recent work [13,22] already showed the positive effect of WWM in pre-training on the performance of the downstream task. In this article, we also examine the differences between the original MLM task and the MLM task with WWM.

4. Fine-Tuning Techniques for NER

The task of NER is to detect entities, such as persons or places, which possibly consist of several words within a text. As proposed in [3], the traditional approach for fine-tuning a BERT to a classification task, such as NER, is to attach an additional feed-forward layer to a pre-trained BERT that predicts token-wise labels. In order to preserve and obtain information about the grouping of tokens into entities, Inside-Outside-Beginning (IOB) tagging [23] is usually applied. IOB tagging introduces two versions of each entity class, one marking the beginning of the entity and one representing the interior of an entity, and an "other" class, which all together results in a total of $\gamma = 2e + 1$ tag classes, where e is the number of entity classes. Table 3 shows an example in which the beginning token of an entity is prefixed with a "B-" and all other tokens with an "I-".

Table 3. IOB tagging example with unlabeled words (O) and the two entities: "location" (Loc) and "person" (Per). The first tag of each entity is prefixed with "B-", while all the following tokens of that entity are marked with an "I-". The first row are the words of the sentence that are split into one or more tokens (second row). The third row shows the tagged tokens based on the given entities (last row). The example sentence can be translated as "Peter lives in Frankfurt am Main".

Words	Peter	lebt	in	Frankfurt		am	Main
Tokens	Peter	lebt	in	Frank	_furt	am	Main
Tagged Tokens	B-Per	O	O	B-Loc	I-Loc	I-Loc	I-Loc
Entities	Person			Location			

In compliance with the standard evaluation scheme of NER tasks in [24], we compute an entity-wise F_1 score denoted by E-F_1. Instead of computing a token- or word-wise F_1 score, E-F_1 evaluates a complete entity as true positive only if all tokens belonging to the entity are correct. Our implementation of E-F_1 relies on the widely used Python library *seqeval* [25].

Usually, IOB tagging is trained by a token-wise softmax cross-entropy loss. However, this setup of one feed-forward layer and a cross-entropy loss does not take into account the context of the tokens forming an entity. In the following, we will call this default approach of fine-tuning the BERT Default-Fine-Tuning. It can lead to inconsistent tagging; for example, an inner tag may only be preceded by an inner or beginning tag of the same entity and thus results in a devastating impact on the E-F_1-score. Therefore, we propose and compare three modified strategies that include context to prevent inconsistent NER tagging during training or decoding. The first approach is a modification of the IOB tagging, the second proposal uses Linear Chain Conditional Random Fields (LCRFs), and the last attempt applies rules to fix a predicted tagging.

Most papers on BERT models dealing with German NER, for example [13] or [12], do not focus on an investigation of different variants for fine-tuning. However, there are already studies for NER tasks in other languages (e.g., [26,27]) that show that the application of LCRFs can be beneficial for fine-tuning. The authors of [27] also investigated whether it is advantageous for the fine-tuning of BERT models on NER tasks to link the pre-trained BERT models with LSTM layers. However, these experiments did not prove to be successful.

4.1. Fine-Tuning with CSE Tagging

In this section, we propose an alternative to the IOB tagging that we call Class-Start-End (CSE) tagging. The main idea is to split the task into three objectives, as shown in Table 4: finding start and end tokens and learning the correct class.

Table 4. CSE tagging example. Rows refer to the tokens and its respective target for start, end, and class.

Tokens	Peter	lebt	in	Frank	_furt	am	Main
Start	1	0	0	1	0	0	0
End	1	0	0	0	0	0	1
Class	Per	O	O	Loc	Loc	Loc	Loc

CSE appends two additional dense layers with logistic-sigmoid activation to the last BERT layer with scalar outputs, one for the start p^{start} and one for the end p^{end} token. In summary, the complete output for an input sample consisting of n tokens is $\left((p_1^{\text{start}}, p_1^{\text{end}}, y_1), (p_2^{\text{start}}, p_2^{\text{end}}, y_2), \ldots, (p_n^{\text{start}}, p_n^{\text{end}}, y_n)\right) \in \mathbb{R}^{n\times(2+e+1)}$, where $e+1$ is the number of possible entities and the "other" class.

The objective for y_i is trained with softmax cross-entropy as before but without the distinction between B- and I-, while the start and end vectors contribute extra losses J^{start} and J^{end}:

$$J^{\text{start}} = -\sum_{j=1}^{n} \left[t_j^{\text{start}} \cdot \log\left(p_j^{\text{start}}\right) + \left(1 - t_j^{\text{start}}\right) \cdot \log\left(1 - p_j^{\text{start}}\right)\right], \tag{5}$$

where t^{start} and p^{start} are the target and prediction vectors for start, as shown in Table 4. J^{end} is the defined analog.

Converting the CSE into IOB tagging is realized by accepting tokens that exceed the threshold of 0.5 as start or end markers. If an end marker is missing between two start markers, the position of the highest end probability between the two locations is used as an additional end marker. This approach is applied analogue in reverse to miss the start markers. Finally, all class probabilities between each start and end marker pairs (including start and end) is averaged to obtain the entity class. In conclusion, an inconsistent tagging is impossible.

4.2. Fine-Tuning with Linear Chain Conditional Random Field with NER-Rule (LCRF$_{NER}$)

Another approach to tackle inconsistent IOB tagging during fine-tuning of a BERT is based on Linear Chain Conditional Random Fields (LCRFs), which are a modification of Conditional Random fields, both proposed in [28]. LCRFs are a common approach to train neural networks that model a sequential task and are therefore well suited for fine-tuning

NER. The basic idea is to take into account the classification of the neighboring sequence members when classifying an element of a sequence.

The output $Y = (y_1, y_2, \dots, y_n) \in \mathbb{R}^{n\times\gamma}$ of our neural network for the NER task consists of a sequence of n vectors whose dimension corresponds to the number of classes $\gamma \in \mathbb{N}$. LCRF introduce so-called transition values $\mathfrak{T}$ that are a matrix $W^{\mathfrak{T}}$ of trainable weights, in the basic approach: $\mathfrak{T} := W^{\mathfrak{T}} \in \mathbb{R}^{\gamma\times\gamma}$. An entry $\mathfrak{T}_{i,j}$ of this matrix $\mathfrak{T}$ can be seen as the potential that a tag of class i is followed by a tag of class j. In one of the easiest forms of LCRFs, which we choose, decoding aims to find the sequence $C^p := \left\{c_1^p, c_2^p, \dots, c_n^p\right\} \in \{1, 2, \dots, \gamma\}^n$ with the highest sum of corresponding transition values and elements of the corresponding output vectors, as shown in Equation (6).

$$C^p := \underset{C\in\{1,\dots,\gamma\}^n}{\arg\max} \left(\sum_{j=1}^{n} y_{j,c_j} + \sum_{j=1}^{n-1} \mathfrak{T}_{c_j,c_{j+1}} \right) \tag{6}$$

Equation (6) is efficiently solved by the Viterbi-Algorithm (see, e.g., [29]). During training, a log-likelihood loss is calculated that takes into account the transition values $\mathfrak{T}$ and the network output Y. The authors in [29] provide a detailed description for its implementation.

Since the IOB tagging does not allow all possible transitions, [30] tried to simply ban these forbidden transitions completely by assigning fixed non-trainable high negative values to the associated entries in $\mathfrak{T}$. However, this did not lead to any improvement in performance, but they were able to show that this allows fine-tuning to converge faster when switching from the classic IOB tagging to the more detailed IOBES tagging scheme [30]. In contrast to them, we extend the original LCRF approach by explicitly modeling these forbidden transitions by adding additional trainable weights to the model when computing the transition values $\mathfrak{T}$. In the following, we call our adapted algorithm LCRF$_{\text{NER}}$.

Assume an NER task comprises the set of entities $X_1, X_2, \dots, X_e$, which results in $\gamma = 2e + 1$ classes following the IOB tagging scheme. Thus, beside a label O for unlabeled elements, for each entity X$_i$, there is a begin label B-X$_i$ and an inner label I-X_i. For simplicity, we order these classes by B-X$_1$, …, B-X$_e$, I-X$_1$, …, I-X$_e$, O, that is:

$$\text{Class } i \text{ belongs to label} \begin{cases} \text{B-X}_i & \text{if } i \le e \\ \text{I-X}_{i-e} & \text{if } e < i \le 2e \\ O & \text{otherwise.} \end{cases}$$

With respect to this ordering, we introduce the matrix $\mathfrak{F} \in \{0,1\}^{\gamma\times\gamma}$ of all forbidden transitions as

$$\mathfrak{F}_{i,j} = \begin{cases} 1 & \text{if } e < j \le 2e \text{ and } i \ne j \text{ and } i \ne j - e \\ 0 & \text{otherwise.} \end{cases}$$

Thus, an element $\mathfrak{F}_{i,j}$ is 1, if and only if a tag of class j can not follow on a tag of class i in the given NER task. This maps the constraint that the label of the predecessor of an interior tag of label I-X can only be the same interior label I-X or the corresponding begin label B-X.

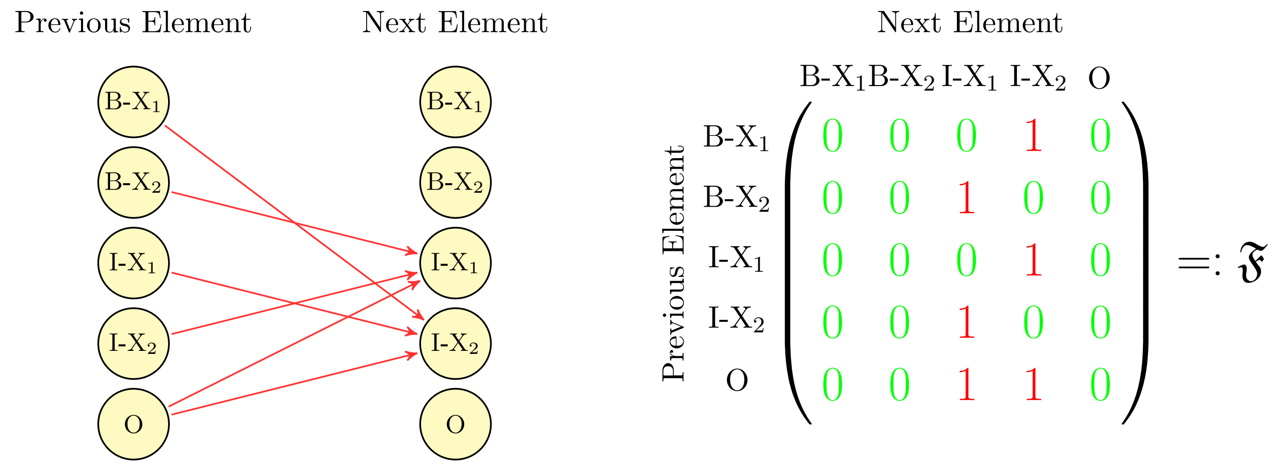
In Figure 1, we illustrate the definition of $\mathfrak{F}$.

Likewise, we define the matrix $\mathfrak{A} \in \{0,1\}^{\gamma\times\gamma}$ by $\mathfrak{A}_{i,j} = 1 - \mathfrak{F}_{i,j}$ as the matrix of all allowed tag transitions. LCRF$_{\text{NER}}$ introduces two additional trainable weights $\omega^{\mathfrak{F}}_{\text{factor}}, \omega^{\mathfrak{F}}_{\text{absolute}} \in \mathbb{R}$ besides the weights $W^{\mathfrak{T}}$ and constructs $\mathfrak{T}$ by

$$\mathfrak{T} := (\mathfrak{A} + \omega^{\mathfrak{F}}_{\text{factor}}\mathfrak{F}) \odot W^{\mathfrak{T}} - \omega^{\mathfrak{F}}_{\text{absolute}}\mathfrak{F}, \tag{7}$$

where $\odot$ is the point-wise product. If setting $\omega^{\mathfrak{F}}_{\text{factor}} = 1$ and $\omega^{\mathfrak{F}}_{\text{absolute}} = 0$, this defaults to the original LCRF approach. In this way, the model can learn an absolute penalty by

$\omega_{\text{absolute}}^{\mathfrak{F}}$ and a relative penalty by $\omega_{\text{factor}}^{\mathfrak{F}}$ for forbidden transitions. Note that LCRF$_{\text{NER}}$ is mathematically equivalent to LCRF; the only purpose is to simplify and to stabilize the training.



$$\begin{array}{c|ccccc} & \text{B-X}_1 & \text{B-X}_2 & \text{I-X}_1 & \text{I-X}_2 & \text{O} \\ \hline \text{B-X}_1 & 0 & 0 & 0 & 1 & 0 \\ \text{B-X}_2 & 0 & 0 & 1 & 0 & 0 \\ \text{I-X}_1 & 0 & 0 & 0 & 1 & 0 \\ \text{I-X}_2 & 0 & 0 & 1 & 0 & 0 \\ \text{O} & 0 & 0 & 1 & 1 & 0 \end{array} =: \mathfrak{F}$$

Figure 1. Example for the definition of the matrix $\mathfrak{F}$ of all forbidden transitions for two entities X_1, X_2. If we follow the IOB tagging scheme, red arrows mark forbidden transitions between two sequence elements that lead to an entry 1 in $\mathfrak{F}$.

4.3. Decoding with Entity-Fix Rule

Finally, we propose a rule-based approach to resolve inconsistent IOB tagging, which can, for example, occur if an I-X tag is subsequent to a token that is not I-X or B-X (for any possible entity class X). Our so-called Entity-Fix rule replaces forbidden I-X tags with the tag of the previous token. If the previous token has a B-X tag, the inserted token is converted to the corresponding I-X tag. In the special case where an I-X tag is predicted at the start of the sequence, it is converted to B-X of the same class. See Table 5 for an example. The advantage of this approach is that it can be applied as a post-processing step independent of training. Furthermore, since only tokens that already form an incorrect entity are affected by this rule, the E-F_1 score can never decrease by applying it. Note that this does not necessarily hold for the token-wise F_1 score, though.

Table 5. Example for the Entity-Fix rule. Rows refer to the tokens, its respective target, prediction, and the prediction resulting from decoding with the Entity-Fix rule. Changes are emphasized in bold.

Tokens	Peter	lebt	in	Frank	_furt	am	Main
Target	B-Per	O	O	B-Loc	I-Loc	I-Loc	I-Loc
Prediction	**I-Per**	O	O	B-Loc	**I-Org**	**I-Org**	I-Loc
Prediction with Fix-Rule	**B-Per**	O	O	B-Loc	**I-Loc**	**I-Loc**	I-Loc

5. BERT Architecture with Whole-Word Attention (WWA)

In this section, we describe our proposed word-wise attention layers used by some of our BERT models during pre-training and fine-tuning. This Whole-Word Attention (WWA) was inspired by the benefits of the Whole-Word Masking (WWM). It comprises two components: the first one called mha$_{\text{wwa}}$ applies traditional multi-head attention on words instead of tokens, while the second component is a windowed attention module called mha$_{\text{wind}}$.

5.1. Traditional Approach

In contrast to current NLP network architectures, previous approaches for tokenizing text (e.g., [31]) did not apply a tokenizer to break down each word of a sentence into more than one possible token. Instead, they trained representations for a fixed vocabulary of words. The major drawback was that this required a large vocabulary, and out-of-vocabulary words could not be represented. Modern approaches tokenize words by a vocabulary of subwords that allows composing unknown words by known tokens. However, when combined with Transformers, attention is computed between pairs of tokens. As a consequence, the number of energy values (see Equation (1)) to be calculated increases quadratically with sequence length, resulting in a large increase in memory and computation time for long sequences.

Different approaches exist to tackle this problem. The most prominent ones are BigBird [32] and Longformer [33]. In their work, the focus is on pure sparse attention strategies: Instead of a full attention, they try to omit as many calculations of energy values as possible, so as little performance as possible is lost. Instead, we propose to rejoin tokens into word-based tokens that also have a quadratic dependence on the sequence length but by a lower slope.

5.2. Our Methodology

The purpose of the first module, mha_{wwa}, is to map tokens back to words and then to compute a word-wise attention. However, since mha_{wwa} loses information about the order of tokens within a word, we introduce mha_{wind} as an additional component that acts on the original tokens. mha_{wind} scales linearly with the sequence length since only a window of tokens is taken into account when computing the energy vectors. In summary, mha_{wwa} learns the global coarser dependence of words, whereas mha_{wind} allows resolving and learning relations of tokens but only in a limited range. In the following, we first describe mha_{wwa} and then mha_{wind}.

Let T denote the input of our BERT model, which is a part of text and can thus be seen as a sequence of words $T = (w_1, w_2, \ldots, w_m)$ with $m \in \mathbb{N}$. Similar to a classical BERT, a tokenizer $\mathcal{T}$ transforms T into a sequence of tokens $\mathcal{T}(T) =: t = (t_1, t_2, \ldots, t_n) \in \mathbb{N}^n$ with $m \leq n$ because we only consider traditional tokenizers that encode the text word-wisely by decomposing a word into one or more tokens. Such a tokenizer provides a mapping function $F_{T,\mathcal{T}} : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, m\}$ that uniquely maps an index i of the token sequence t to the index j of its respective word w_j.

Each encoder layer ℓ of the classical BERT architecture contains a multi-head attention layer mha^ℓ, which maps its input sequence $X^\ell_{T,\mathcal{T}} = (x^\ell_1, x^\ell_2, \ldots, x^\ell_n) \in \mathbb{R}^{n \times d}$ to an output $Y^\ell_{T,\mathcal{T}}$ of equal length n and dimension d:

$$\text{mha}^\ell(X^\ell_{T,\mathcal{T}}) = Y^\ell_{T,\mathcal{T}} = (y^\ell_1, y^\ell_2, \ldots, y^\ell_n) \in \mathbb{R}^{n \times d},$$

where the ith output vector y^ℓ_i is defined as the concatenation of the resulting vectors for every attention head computed by Equation (2). Our mha_{wwa} layer modifies this by applying attention only on the sequence $\hat{X}^\ell_{T,\mathcal{T}} = (\hat{x}^\ell_1, \hat{x}^\ell_2, \ldots, \hat{x}^\ell_m) \in \mathbb{R}^{m \times d}$, where

$$\hat{x}_j := \frac{1}{|\{i : F_{T,\mathcal{T}}(i) = j\}|} \sum_{i : F_{T,\mathcal{T}}(i) = j} x_i \tag{8}$$

and $\{i : F_{T,\mathcal{T}}(i) = j\}$ is the set of all tokens i belonging to the word j. In other words, we average the corresponding token input vectors for each word. Next, we apply mha on $\hat{X}^\ell_{T,\mathcal{T}}$ yielding the output

$$\text{mha}^\ell(\hat{X}^\ell_{T,\mathcal{T}}) =: \hat{Y}^\ell_{T,\mathcal{T}} = (\hat{y}^\ell_1, \hat{y}^\ell_2, \ldots, \hat{y}^\ell_m) \in \mathbb{R}^{m \times d}$$

which is a sequence of length m only. Finally, to again obtain a sequence of length n, we transform the output sequence back to the length n by repeating the output vector for each word according to the number of associated tokens. Thus, the final output of a layer $\text{mha}^\ell_{\text{wwa}}$ is defined as

$$\text{mha}^\ell_{\text{wwa}}(X^\ell_{T,\mathcal{T}}) := Z^\ell_{T,\mathcal{T}} = (z^\ell_1, z^\ell_2, \ldots, z^\ell_n) \in \mathbb{R}^{n \times d}$$

where $z^\ell_i := \hat{y}^\ell_{F_{T,\mathcal{T}}(i)}$. See Figure 2 for an illustration of the concept described above.

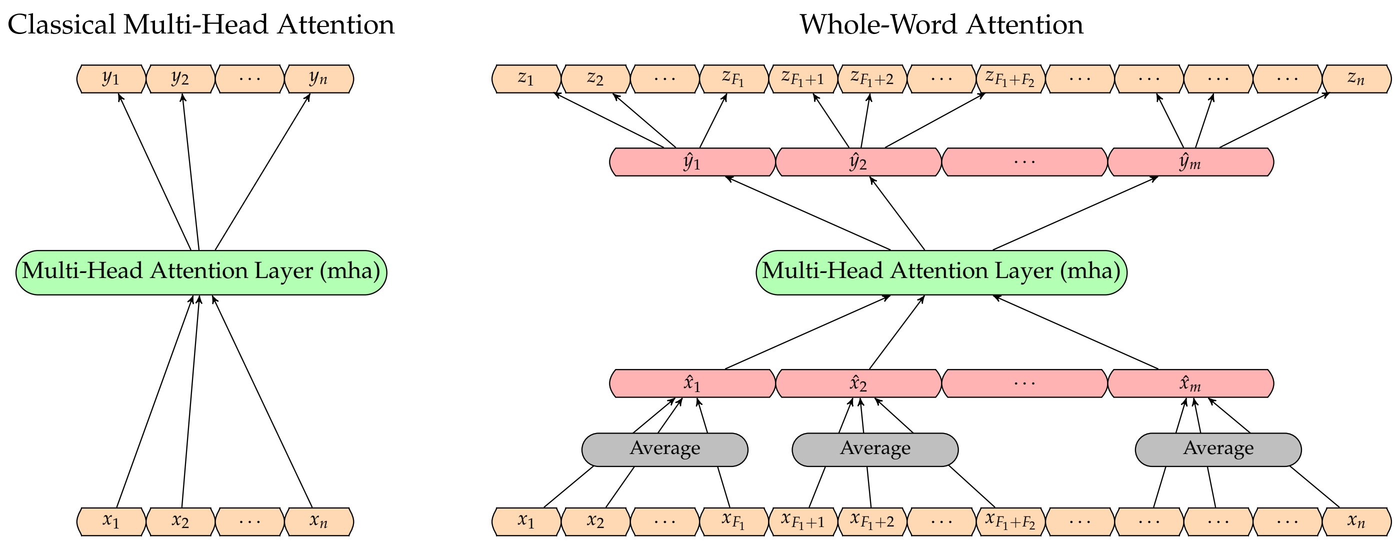

Classical Multi-Head Attention
Whole-Word Attention
y_1 y_2 $\cdots$ y_n
Multi-Head Attention Layer (mha)
x_1 x_2 $\cdots$ x_n
z_1 z_2 $\cdots$ z_{F_1} z_{F_1+1} z_{F_1+2} $\cdots$ $z_{F_1+F_2}$ $\cdots$ z_n
$\hat{y}_1$ $\hat{y}_2$ $\cdots$ $\hat{y}_m$
$\hat{x}_1$ $\hat{x}_2$ $\cdots$ $\hat{x}_m$
Average
x_1 x_2 $\cdots$ x_{F_1} x_{F_1+1} x_{F_1+2} $\cdots$ $x_{F_1+F_2}$ $\cdots$ x_n

Figure 2. Left: Multi-Head Attention. **Right**: Our concept of Whole-Word Attention, where classical Multi-Head Attention is applied on words instead of tokens. Orange: Members of sequences, whose length is the number of tokens n. Red: Members of sequences, whose length is the number of words m. For a better overview, we define $F_j := \left|\{i : F_{T,\mathcal{T}}(i) = j\}\right|$ as the number of tokens the j'th word consists of.

We perform positional encoding by utilizing relative positional encoding because absolute positional encoding adds the positional vectors to the token sequence directly after the embedding. This is not compatible with WWA of Equation (8). Instead, relative positional encoding adds a word-wise relative positional encoding to the vectors of the word-wise sequence within $\text{mha}^\ell(\hat{X}^\ell_{T,\mathcal{T}})$.

As an experiment, we also pre-trained a BERT that solely uses mha_{wwa} layers for attention. However, it was already apparent in the pre-training that it could only achieve a very low Mask Language Model (MLM) accuracy. The main reason for this is that $Z^\ell_{T,\mathcal{T}}$ does not take into account any information about the position of the tokens within a word since the output elements of them are equal, which is why these tokens are no longer related to each other via attention. To tackle this problem, for each ℓ, we introduce a second multi-head attention layer $\text{mha}^\ell_{\text{wind}}$ based on windowed attention as used in [32,33]. Because the sole purpose of $\text{mha}^\ell_{\text{wind}}$ is to map the relationships and positions of the tokens within a word in the model, we use a very small sliding window size of $\omega = 5$ tokens in each direction. Hence, in contrast to [32,33], we also do not arrange our input sequence into blocks or chunks.

Formally, we define mha_{wind} as

$$\text{mha}^\ell_{\text{wind}}(X^\ell_{T,\mathcal{T}}) := Y^\ell_{T,\mathcal{T}} = (y^\ell_1, y^\ell_2, \ldots, y^\ell_n) \in \mathbb{R}^{n\times d}$$

where the ith output vector y^ℓ_i is the concatenation of the resulting vectors $y^{\ell,h}_i$ for every attention head h. Each $y^{\ell,h}_i$ is given by

$$y^{\ell,h}_i = \sum_{j=i-\omega}^{i+\omega} \alpha^h_{i,j}\left(W^{V,h}x_j + d^{V,h}_{j-i}\right)$$

where $\alpha^h_{i,j} = \frac{exp\left(e^h_{i,j}\right)}{\sum\limits_{k=1}^{n} exp\left(e^h_{i,k}\right)}, W^{V,h} \in \mathbb{R}^{d_{model}\times d_v}$. The energy values $e^h_{i,j}$ for each attention head h are defined as in Equation (1), and the distance vectors $d^{V,h}_{j-i}, d^{K,h}_{j-i}$ are given by the Equations (3) and (4).

In summary, in our BERT model using WWA, the total output of the ℓ-th attention layer of the ℓ-th encoder layer is, after adding the input sequence $X^{\ell}_{T,\mathcal{T}}$ as a residual,

$$X^{\ell}_{T,\mathcal{T}} + \mathrm{mha}^{\ell}_{\mathrm{wwa}}(X^{\ell}_{T,\mathcal{T}}) + \mathrm{mha}^{\ell}_{\mathrm{wind}}(X^{\ell}_{T,\mathcal{T}})$$

compared to the original approach

$$X^{\ell}_{T,\mathcal{T}} + \mathrm{mha}^{\ell}(X^{\ell}_{T,\mathcal{T}}).$$

$\mathrm{mha}_{\mathrm{wind}}$ introduces additional trainable variables compared to the traditional Transformer architecture. However, the number of energy values only increases linearly compared to the sequence length with respect to the window size ω by a factor $2\omega + 1$. Hence, the impact on the memory can be neglected for long sequences and s$m$all ω. In order to quantify the overall reduction of memory consumption using WWA, we provide an example using our tokenizer built on the German Wikipedia with a vocabulary of about 30,000, as is common in many BERT models. It transforms on average a sequence of m words in $n \approx 1.5m = \frac{3}{2}m$ tokens. Therefore, while the traditional multi-head attention layer calculates n^2 energy values per each head, our WWA approach only requires

$$\overbrace{m^2}^{\mathrm{mha}_{\mathrm{wwa}}} + \overbrace{(2\omega+1)n}^{\mathrm{mha}_{\mathrm{wind}}} \approx \frac{4}{9}n^2 + (2\omega+1)n \tag{9}$$

energy values. Thus, for large n (and small ω), the number of energy values to be calculated is more than halved.

6. Experiments

To evaluate our proposed methods for German NER tasks, we conducted several experiments. First, we compared the pre-training variants presented in Section 3 and then fine-tuning techniques presented in Section 4. Afterwards, we applied Whole-Word Attention (WWA) on the overall training of the NER task. Finally, we discuss our results in relation to the state-of-the-art models of the LER and GermEval tasks.

6.1. Comparing Pre-Training Techniques

In our first experiments, we investigated which of the known pre-training techniques (see Section 3) yields the best models for a subsequent fine-tuning on German NER tasks. For this purpose, we combined the three presented pre-training tasks (Mask Language Model (MLM), MLM with Sentence Order Prediction (SOP), and MLM with Next Sentence Prediction (NSP)) with relative and absolute positional encoding and optionally enabled Whole-Word Masking (WWM). For each resulting combination, we pre-trained a (small) BERT with a hidden size of 512 with 8 attention heads on 6 layers for 500 epochs and 100,000 samples per epoch. Pre-training was performed with a batch size of 48 and a maximal sequence length of 320 tokens, which is limited by the 11 GB memory of one GPU (RTX 2080 Ti). Each of the 12 resulting BERTs was then fine-tuned using the Default-Fine-Tuning approach (described in Section 4) on the five German NER tasks described in Section 2.2. For fine-tuning, we chose a batch size of 16 and trained by default for 30 epochs and 5000 samples per epoch. The number of epochs was increased to 50 for the larger LER tasks. Each fine-tuning run was performed three times, and its average result was reported in Table 6 together with its standard deviation σ.

Table 6. Columns refer to the average E-F_1 score (cf. [24]) of three fine-tuning runs with Default-Fine-Tuning (see Section 4) for five datasets and its standard deviation σ multiplied by 100. Rows refer to the respective pre-training task, absolute or relative positional encoding (PE), and use of Whole-Word Masking (WWM); best results within $2 \cdot \sigma$ of the maximum (best) are emphasized

Pre-Train Task	PE	WWM	GermEval		H.Arendt		Sturm		LER CG		LER FG		Average
			E-F_1	100σ	E-F_1	100σ	E-F_1	100σ	E-F_1	100σ	E-F_1	100σ	E-F_1
MLM	abs.	-	0.7785	0.09	0.7600	1.28	0.8236	1.20	0.9061	0.48	0.8969	0.37	0.8330
MLM	abs.	✓	0.7901	0.21	0.7681	0.40	0.8102	1.73	0.9192	0.12	0.9028	0.21	0.8381
MLM	rel.	-	0.7852	0.16	0.7674	0.54	0.8011	1.45	0.9198	0.34	0.9144	0.27	0.8376
MLM	rel.	✓	**0.8086**	0.24	**0.7741**	0.79	**0.8555**	0.34	**0.9347**	0.30	**0.9206**	0.14	**0.8587**
MLM, NSP	abs.	-	0.7609	0.10	0.7688	1.21	0.8085	0.20	0.9067	0.36	0.8928	0.56	0.8275
MLM, NSP	abs.	✓	0.7484	1.52	0.7621	0.40	0.8110	1.38	0.9047	0.30	0.8930	0.30	0.8238
MLM, NSP	rel.	-	0.7802	0.40	0.7471	0.48	0.8172	1.37	0.9193	0.39	0.9105	0.18	0.8348
MLM, NSP	rel.	✓	0.7750	0.23	0.7564	1.15	0.8191	0.40	0.9168	0.07	0.9044	0.38	0.8343
MLM, SOP	abs.	-	0.7530	0.18	0.7669	0.71	0.7942	1.20	0.8979	0.25	0.8817	0.14	0.8188
MLM, SOP	abs.	✓	0.7355	0.38	0.7590	0.21	0.7995	2.40	0.9047	0.17	0.8950	0.35	0.8187
MLM, SOP	rel.	-	0.7745	0.58	0.7548	0.52	0.8052	0.62	0.9176	0.24	0.9040	0.48	0.8312
MLM, SOP	rel.	✓	0.7863	0.31	**0.7807**	0.41	**0.8550**	0.15	0.9246	0.24	0.9122	0.38	0.8518

The first thing we can see in Table 6 is that, as expected, the standard deviation for the NER tasks with a smaller ground truth (mainly Sturm but also H.Arendt) is higher than for NER tasks with larger ground truth (GermEval, both LER variants). The fluctuations of the different fine-tuning runs are nevertheless within a reasonable range.

The results in Table 6 reveal that the best performing BERTs for NER tasks used relative positional encoding, WWM, and were solely pre-trained with MLM. This is plausible since the samples, i.e., the academical NER datasets, are only single sentences and hence using additional sentence-spanning losses during pre-training (SOP or NSP) is even harmful for these downstream tasks.

Furthermore, our experiments show that relative positional encoding yields significantly better results than absolute positional encoding. This is an interesting observation since the experiments from [20] stated that relative positional encoding only performs better when applied to sequences with longer lengths than those on which the network was previously trained. To investigate this in detail, we performed an analysis of the E-F_1 score in dependence of the sequence length. We sorted the samples in the test list for each dataset by token length in increasing order and then split them into seven parts of equal number of samples. The left chart of Figure 3 shows the averaged E-F_1 score of the five datasets for each part. We observe that relative encoding (rel) is outperforming absolute encoding (abs) for almost any sequence length. For long sequences (last part), the gap between the two approaches increases, which is expected. The right chart of Figure 3 shows the E-F_1 score for each part of only the H. Arendt dataset. For this dataset, the discrepancy between relative and positional encoding is very small, which shows that the benefit of relative positional encoding highly depends on the dataset. Nevertheless, on average, our experiments suggest using relative positional as the mean of choice for German NER tasks.

Furthermore, we observe that WWM together with relative positional encoding led to significant improvements when combined with MLM with or without SOP as pre-training task. In summary, our best setup combined solely MLM, relative positional encoding, and WWM.

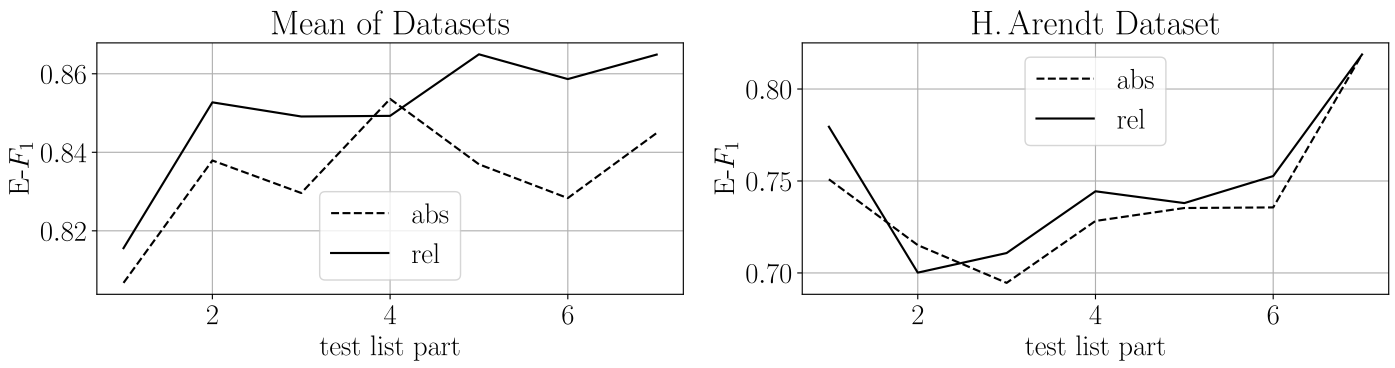


Figure 3. The E-F_1 score for a BERT trained on Mask Language Model (MLM) task with Whole-Word Masking (WWM) for relative and absolute positional encoding fine-tuned with Default-Fine-Tuning (see Section 4) tested on seven parts of the test list, which are sorted by token length, starting with short sequences. **Left** shows the mean over all datasets, and the **right** shows the results on one dataset (H. Arendt).

6.2. Comparing Fine-Tuning Techniques

In this section, we examine our four different variants of fine-tuning: Default-Fine-Tuning (see Section 4), Class-Start-End (CSE) (see Section 4.1), and Linear Chain Conditional Random Field (LCRF), or LCRF$_{\text{NER}}$ (see Section 4.2). The new fine-tuning techniques, CSE and LCRF$_{\text{NER}}$, were specifically designed to help the network learn the structure of IOB tagging.

First, we evaluated the impact of the fine-tuning method in dependence of the pre-training techniques examined in Section 6.1. For this purpose, we fine-tuned all BERT models in analogy to Section 6.1 on all five NER tasks using the four fine-tuning methods mentioned. The individual results shown in Table 7 are averaged across the five tasks, whereby the outcome of each task is the mean of three runs.

Table 7. Average E-F_1 score of all five NER tasks and three fine-tuning runs. Rows refer to the respective pre-training task, absolute or relative positional encoding (PE), and use of Whole-Word Masking (WWM); Columns refer to the fine-tuning methods Default-Fine-Tuning (DFT), Class-Start-End (CSE), Linear Chain Conditional Random Field (LCRF) and LCRF$_{\text{NER}}$; best results per column are emphasized.

Pre-Training Task	PE	WWM	Avg. DFT	Avg. CSE	Avg. LCRF	Avg. LCRF$_{\text{NER}}$
MLM	abs.	-	0.8330	0.8608	0.8455	0.8469
MLM	abs.	✓	0.8381	0.8635	0.8443	0.8516
MLM	rel.	-	0.8376	0.8682	0.8477	0.8539
MLM	rel.	✓	**0.8587**	**0.8785**	**0.8623**	**0.8651**
MLM, NSP	abs.	-	0.8275	0.8536	0.8346	0.8418
MLM, NSP	abs.	✓	0.8238	0.8543	0.8358	0.8416
MLM, NSP	rel.	-	0.8348	0.8639	0.8457	0.8471
MLM, NSP	rel.	✓	0.8343	0.8595	0.8417	0.8459
MLM, SOP	abs.	-	0.8188	0.8515	0.8292	0.8325
MLM, SOP	abs.	✓	0.8187	0.8597	0.8312	0.8369
MLM, SOP	rel.	-	0.8312	0.8598	0.8450	0.8473
MLM, SOP	rel.	✓	0.8518	0.8714	0.8557	0.8605
Overall Average			0.8340	0.8608	0.8432	0.8476

Regardless of the choice of the fine-tuning method, the results confirm that pre-trained BERTs that were only pre-trained with MLM use relative Positional Encoding and use WWM are the most suitable for German NER tasks. Therefore, we will only consider this fine-tuning setup in the following.

Furthermore, the results show that LCRF$_{\text{NER}}$ consistently outperforms LCRF. We think that the introduction of the new weights $\omega^{\mathfrak{F}}_{\text{factor}}, \omega^{\mathfrak{F}}_{\text{absolute}}$ (see Equation (7)) enables the fine-tuning with LCRF$_{\text{NER}}$ to outperform LCRF because the network can more easily learn to avoid inconsistent tagging. However, CSE performs best on average. We suspect that this is due to the fact that the way of decoding in CSE completely prevents inconsistent tagging.

Table 8 provides more details for the best setup by listing separate results per NER task. Furthermore, we examined the influence of applying the rule-based Entity-Fix (see Section 4.3) as a post-processing step.

Table 8. Average E-F_1 score of three fine-tuning runs on BERTs pre-trained with Mask Language Model (MLM), relative positional encoding, and Whole-Word Masking (WWM). Rows refer to the fine-tuning methods Default-Fine-Tuning (DFT), Class-Start-End (CSE), Linear Chain Conditional Random Field (LCRF), and LCRF$_{\text{NER}}$; Columns refer to the NER task; best results per column are emphasized.

Fine-Tuning Task	Entity-Fix Rule	GermEval	H. Arendt	Sturm	LER CG	LER FG	Average
DFT	-	0.8086	0.7741	0.8555	0.9347	0.9206	0.8587
DFT	✓	0.8408	0.7903	0.8706	0.9474	0.9427	0.8783
CSE	-	0.8397	**0.8048**	0.8647	0.9429	0.9401	0.8785
CSE	✓	0.8397	**0.8048**	0.8647	0.9429	0.9401	0.8785
LCRF	-	0.8216	0.7822	0.8453	0.9365	0.9261	0.8623
LCRF	✓	0.8422	0.7941	0.8629	0.9477	0.9410	0.8776
LCRF$_{\text{NER}}$	-	0.8220	0.7857	0.8508	0.9394	0.9278	0.8651
LCRF$_{\text{NER}}$	✓	**0.8448**	0.7999	**0.8783**	**0.9488**	**0.9455**	**0.8823**

The Entity-Fix rule was specifically designed to fix inconsistencies that otherwise would lead to an error on the metric. Since the CSE decoding already includes the rules of the metric, no changes are present if applying the Entity-Fix rule. This post-processing step led to clear improvements on all other fine-tuning methods. Surprisingly, although LCRF and LCRF$_{\text{NER}}$ were specially designed to learn which consecutive tags were not allowed in the sequence, they still show significant improvements with the Entity-Fix rule. Thus, they were not able to learn the structure of the IOB tagging scheme sufficiently.

The general result is that LCRF$_{\text{NER}}$ represents the best fine-tuning method if combined with the Entity-Fix rule.

Additionally, we examined why, in contrast to all other tasks, the H. Arendt task with CSE yielded slightly better results than with LCRF$_{\text{NER}}$ and the Entity-Fix rule by comparing the errors made on the test set. Unfortunately, no explanation could be found in the data. Nevertheless, LCRF$_{\text{NER}}$ emerges as the best fine-tuning method from our experiments in general.

6.3. Results of Whole-Word Attention (WWA)

Next, we investigated the influence of replacing the original multi-head attention layers with our proposed Whole-Word Attention (WWA) approach. For this, we pre-trained BERTs with only the Mask Language Model (MLM), relative positional encoding, and Whole-Word Masking (WWM) with the same hyper-parameter setup as in Section 6.1. Since, as shown in Equation (9), WWA allows increasing the maximal token sequence length, we first pre-trained a BERT with a token sequence length of 320 (similar to the previous experiments) but also one with a maximum sequence length of 300 words, which is roughly 450 tokens on average (see Equation (9)). This value is chosen so that approximately the same number of energy values were calculated in this BERT model as in the comparable model without WWA.

After pre-training of the two BERTs was finished, we fine-tuned all NER tasks with our best fine-tuning method LCRF$_{\text{NER}}$. Table 9 compares the results of using WWA to those without (see Table 8).

Table 9. Average E-F_1 score of three fine-tuning runs with LCRF_{NER} on BERTs pre-trained with Mask Language Model (MLM), relative positional encoding, and Whole-Word Masking (WWM). Rows refer to the use of WWA and the maximal sequence length in pre-training; Columns refer to the use of the Entity-Fix rule.

WWA	Max Seq. Length Pre-Train	Avg. E-F_1	Avg. E-F_1 with Entity-Fix Rule
-	320 token	0.8651	0.8823
✓	320 token	0.8676	0.8832
✓	300 words	0.8674	0.8832

The results show that, on average, WWA slightly improved the original approach, even though training was done with reduced memory consumption due to the smaller number of energy values to be calculated and stored. The drawback is that the pre-training of a BERT with WWA takes about 1.5 times longer than pre-training without WWA due to the additional window attention layer and the transformation of the sequence from token to word and vice versa. The runtime could probably be accelerated by improving the implementation for the transformation from token sequence to word sequence without looping over the samples of the batch.

Increasing the maximum sequence length to 300 words did not result in an additional advantage of the performance. We suspect that the reason is that the maximum sequence length for NER tasks is not a primary concern because the samples in all five NER datasets tested almost never exhausted the maximum sequence length used in pre-training. However, we expect a benefit for other downstream tasks such as document classification or question answering where longer sequences and long-range relations are more important.

6.4. Comparing Results with the State of the Art

In this section, we compare our results with the current state of the art. This is only possible for the two LER tasks and the GermEval task since the other two NER datasets were newly introduced by this paper.

To the best of our knowledge, the current state of the art in both LER tasks was achieved in [34]. They applied a non-Transformer model on the basis of bidirectional LSTM layers and a LCRF with classical pre-trained word embeddings. For evaluation, in contrast to our used E-F_1 score, they took a token-wise F_1 score, thus calculating precision and recall per token.

Table 10 shows the comparison of the token-wise F_1 score to our best results.

Table 10. Comparison of our best results on the two LER-tasks with the previous state of the art models of [34], where T-F_1 and E-F_1 are the token- and entity-wise F_1 scores, respectively.

Task	Model	T-F_1	E-F_1
LER CG	Previous SoTA [34]	0.9595	-
LER CG	our best	0.9842	0.9488
LER FG	Previous SoTA [34]	0.9546	-
LER FG	our best	0.9811	0.9455

Note, however, that the comparison of the token-wise F_1 score is not entirely correct because in our models, words can sometimes break down into several tokens, whereas, in [34], presumably exactly one token is always used per word. Furthermore, since it was not published, we were not able to use the identical splits of train, validation, and test data.

Next, in Table 11, we compare our results for the GermEval task with the current state of the art, which, to the best of our knowledge, was achieved in [13]. In addition, we list the best results that were achieved without a Transformer architecture [14].

Table 11. Comparison of our best results on the GermEval-task with other state of the art models, where E-F_1 is the entity-wise F_1 score. Our shown result is again the average of 3 fine-tuning runs. The score of DistilBERT was taken from https://huggingface.co/dbmdz/flair-distilbert-ner-germeval14 accessed on 19 October 2021.

Model	**Params**	**E-F_1**
BiLSTM-WikiEmb [14]	-	0.8293
DistilBERT$_{\text{Base}}$ [35]	66 mio	0.8563
GBERT$_{\text{Base}}$ [13]	110 mio	0.8798
GELECTRA$_{\text{Large}}$ [13]	335 mio	0.8895
our best on small BERTs	34 mio	0.8448

While our approach outperforms the BiLSTM results of [14], we were not able to reach the results of [13]. One of the reasons is that our BERTs are smaller and also pre-trained on a much smaller text corpus. The DistilBERT [35] with almost 2× the parameters also reaches a higher score. This comes with the drawback that a larger BERT is needed for pre-training. There are some reasons where it is desired to train a BERT from scratch, for example, to enable research such as the Whole-Word Attention or training on a different domain/language. In Table 12, we illustrate some differences of some technical attributes between our BERT models and GBERT$_{\text{Base}}$ of [13]. It shows that our models can be trained with much lower hardware requirements.

Table 12. Comparison of some technical attributes between GBERT$_{\text{Base}}$ (from [13]) and our small BERTs. The operations per training is a coarse estimation based on the theoretic compute power and pre-training time.

	GBERT$_{\text{Base}}$	**Our Small BERTs**
Parameter	110 mio	34 mio
Pre-training hardware	1× TPU v3	1× GPU
Memory	128 GB	11 GB
Compute power	420 TFLOPS	14 TFLOPS
Tokens seen in pre-training	2.6×10^{11}	1.6×10^{10}
Pre-training time	≈7 days	≈9 days
Operations per training	≈254 EFLOP	≈11 EFLOP

In addition, we compared the time needed for a fine-tuning. For a fair comparison, we downloaded the GBERT$_{\text{Base}}$ from Hugging Face and fine-tuned it with the same hyper-parameters as we fine-tuned our models. This resulted in an average time of 50 min for a fine-tuning run on our models and 70 min for a run on GBERT$_{\text{Base}}$.

7. Conclusions and Future Work

In this article, we conducted our research on comparatively small BERT models to address real-world applications with limited hardware resources, making it accessible to a wider audience. We have worked out how to achieve the best results in German NER tasks with smaller BERT models. This simplifies the work of Germanists in the creation of digital editions.

Therefore, we investigated which pre-training method is the most suitable to solve German NER tasks on three standard and two newly introduced (H. Arendt and Sturm) NER datasets. We examined different pre-training tasks, absolute and relative positional encoding, and masking methods. We observed that a BERT pre-trained only on the Mask Language Model (MLM) task combined with relative positional encoding and Whole-Word Masking (WWM) yielded the overall best results on these downstream tasks.

We also introduced two new fine-tuning variants, LCRF$_{\text{NER}}$ and Class-Start-End (CSE), designed for NER tasks. Their investigation, in combination with Default-Fine-Tuning and common Linear Chain Conditional Random Fields (LCRFs), showed that the best pre-training technique of the BERT is independent of the fine-tuning variant. Furthermore, we introduced the Entity-Fix rule for decoding. Our results showed that for most German NER tasks, $\mathrm{LCRF}_{\mathrm{NER}}$ with the Entity-Fix rule delivers the best results, although there are also tasks for which the CSE tagging has a minor advantage.

In addition, our novel Whole-Word Attention (WWA) that modifies the Transformer architecture resulted in small improvements by simultaneously halving the number of energy values to be calculated. For future work, it would be particularly interesting to investigate WWA in connection with other downstream tasks, such as document classification or question answering, where the processing of longer sequences is more important than in NER tasks. Another approach would be to combine WWA with a sparse-attention mechanism such as BigBird [32].

To further simplify the training and application of BERTs for users with only a low technical background, we are currently developing an open source implementation of these optimized models in a user friendly software with a graphical user interface. The goal of this software is to greatly simplify the creation of digital editions by enriching text stored as TEI files with custom NER taggings.

Author Contributions: Conceptualization, J.Z. and K.S.; Data curation, J.Z. and K.S.; Formal analysis, J.Z. and K.S.; Funding acquisition, R.L.; Investigation, J.Z. and K.S.; Methodology, J.Z. and K.S.; Software, J.Z., K.S. and C.W.; Supervision, R.L.; Validation, J.Z. and K.S.; Visualization, J.Z. and K.S.; Writing—original draft, J.Z. and K.S.; Writing—review and editing, J.Z., K.S., C.W. and R.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the European Social Fund (ESF) and the Ministry of Education, Science, and Culture of Mecklenburg-Western Pomerania (Germany) within the project NEISS (Neural Extraction of Information, Structure, and Symmetry in Images) under grant no ESF/14-BM-A55-0006/19.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: We share the datasets (see Section 2) and our source code (https://github.com/NEISSproject/tf2_neiss_nlp/tree/berNer21), which is based on tfaip [8] with the community.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
CSE	Class-Start-End
IOB	Inside-Outside-Beginning
LCRF	Linear Chain Conditional Random Field
LER	Legal Entity Recognition
MLM	Mask Language Model
NER	Named Entity Recognition
NLP	Natural Language Processing
NSP	Next Sentence Prediction
DFT	Default-Fine-Tuning
SOP	Sentence Order Prediction
WWA	Whole-Word Attention
WWM	Whole-Word Masking

References

1. NEISS Project Neuronal Extraction of Information, Structures and Symmetries in Images. Available online: https://www.neiss.uni-rostock.de/en/ (accessed on 22 October 2021).
2. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2017; pp. 5998–6008.

3. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. 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, Minneapolis, MN, USA, 2–7 June 2019; Volume 1, pp. 4171–4186.
4. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. Albert: A lite bert for self-supervised learning of language representations. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
5. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
6. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.R.; Le, Q.V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2019; pp. 5753–5763.
7. Hugging Face. Available online: https://huggingface.co/ (accessed on 22 October 2021).
8. Wick, C.; Kühn, B.; Leifert, G.; Sperfeld, K.; Strauß, T.; Zöllner, J.; Grüning, T. tfaip—A Generic and Powerful Research Framework for Deep Learning based on Tensorflow. *J. Open Source Softw.* **2021**, *6*, 3297. [CrossRef]
9. Attardi, G. WikiExtractor. 2015. Available online: https://github.com/attardi/wikiextractor (accessed on 15 February 2020).
10. Hamborg, F.; Meuschke, N.; Breitinger, C.; Gipp, B. news-please: A Generic News Crawler and Extractor. In Proceedings of the 15th International Symposium of Information Science, Berlin, Germany, 13–15 March 2017; pp. 218–223. [CrossRef]
11. Benikova, D.; Biemann, C.; Kisselew, M.; Pado, S. GermEval 2014 Named Entity Recognition Shared Task: Companion Paper. 2014. Available online: http://nbn-resolving.de/urn:nbn:de:gbv:hil2-opus-3006 (accessed on 10 November 2020).
12. Labusch, K.; Neudecker, C.; Zellhöfer, D. BERT for Named Entity Recognition in Contemporary and Historic German. In Proceedings of the 15th Conference on Natural Language Processing, Erlangen, Germany, 8–11 October 2019; pp. 1–9.
13. Chan, B.; Schweter, S.; Möller, T. German's Next Language Model. *arXiv* **2020**, arXiv:2010.10906.
14. Riedl, M.; Padó, S. A Named Entity Recognition Shootout for German. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; Volume 2: Short Papers; Association for Computational Linguistics: Stroudsburg, PA, USA, 2018; pp. 120–125. [CrossRef]
15. Leitner, E.; Rehm, G.; Moreno-Schneider, J. A Dataset of German Legal Documents for Named Entity Recognition. *arXiv* **2020**, arXiv:2003.13016.
16. Hahn, B.; Breysach, B.; Pischel, C. Hannah Arendt Digital. Kritische Gesamtausgabe. Sechs Essays. 2020. Available online: https://hannah-arendt-edition.net/3p.html (accessed on 22 October 2021).
17. TEI-Consortium. Guidelines for Electronic Text Encoding and Interchange. 2017. Available online: https://tei-c.org/ (accessed on 22 October 2021).
18. Buchholz, S.; Marsi, E. CoNLL-X Shared Task on Multilingual Dependency Parsing. In Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X), New York, NY, USA, 8–9 June 2006; Association for Computational Linguistics: Stroudsburg, PA, USA, 2006; pp. 149–164.
19. Schrade, M.T. DER STURM. Digitale Quellenedition zur Geschichte der internationalen Avantgarde. 2018. Available online: https://sturm-edition.de/id/S.0000001 (accessed on 22 October 2021).
20. Rosendahl, J.; Tran, V.A.K.; Wang, W.; Ney, H. Analysis of Positional Encodings for Neural Machine Translation. In Proceedings of the IWSLT, Hong Kong, China, 2–3 November 2019.
21. Shaw, P.; Uszkoreit, J.; Vaswani, A. Self-attention with relative position representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 1–6 June 2018; Volume 2, pp. 464–468.
22. Cui, Y.; Che, W.; Liu, T.; Qin, B.; Yang, Z.; Wang, S.; Hu, G. Pre-training with whole word masking for chinese bert. *arXiv* **2019**, arXiv:1906.08101.
23. Ramshaw, L.A.; Marcus, M.P. Text Chunking Using Transformation-Based Learning. In *Natural Language Processing Using Very Large Corpora*; Armstrong, S., Church, K., Isabelle, P., Manzi, S., Tzoukermann, E., Yarowsky, D., Eds.; Text, Speech and Language Technology; Springer: Dordrecht, The Netherlands, 1999; pp. 157–176._10. [CrossRef]
24. Sang, E.F.; De Meulder, F. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *arXiv* **2003**, arXiv:cs/0306050.
25. Nakayama, H. Seqeval: A Python Framework for Sequence Labeling Evaluation. 2018. Available online: https://github.com/chakki-works/seqeval (accessed on 22 October 2021).
26. Luoma, J.; Pyysalo, S. Exploring Cross-sentence Contexts for Named Entity Recognition with BERT. *arXiv* **2020**, arXiv:2006.01563.
27. Souza, F.; Nogueira, R.; Lotufo, R. Portuguese Named Entity Recognition using BERT-CRF. *arXiv* **2020**, arXiv:1909.10649.
28. Lafferty, J.; McCallum, A.; Pereira, F. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Proceedings of the International Conference on Machine Learning (ICML), Williamstown, MA, USA, 28 June–1 July 2001.
29. Sutton, C.; McCallum, A. An Introduction to Conditional Random Fields. 2010. Available online: https://homepages.inf.ed.ac.uk/csutton/publications/crftutv2.pdf (accessed on 22 October 2021).
30. Lester, B.; Pressel, D.; Hemmeter, A.; Ray Choudhury, S.; Bangalore, S. Constrained Decoding for Computationally Efficient Named Entity Recognition Taggers. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP, Online, 16–20 November 2020; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 1841–1848. [CrossRef]

31. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 3111–3119.
32. Zaheer, M.; Guruganesh, G.; Dubey, A.; Ainslie, J.; Alberti, C.; Ontanon, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al. Big bird: Transformers for longer sequences. *arXiv* **2020**, arXiv:2007.14062.
33. Beltagy, I.; Peters, M.E.; Cohan, A. Longformer: The long-document transformer. *arXiv* **2020**, arXiv:2004.05150.
34. Leitner, E.; Rehm, G.; Moreno-Schneider, J. Fine-Grained Named Entity Recognition in Legal Documents. In *Semantic Systems. The Power of AI and Knowledge Graphs*; Acosta, M., Cudré-Mauroux, P., Maleshkova, M., Pellegrini, T., Sack, H., Sure-Vetter, Y., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2019; pp. 272–287. [CrossRef]
35. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.