Institute for Natural Language Processing

University of Stuttgart
Pfaffenwaldring 5B
D–70569 Stuttgart

Bachelorarbeit

Evaluation of deep learning methods for German named entity recognition

Pascal Huszár

Course of Study:	Medieninformatik
Examiner:	Prof. Dr. Thang Vu
Supervisor:	Matthias Blohm, M.Sc. Claudia Dukino, M.Sc.
Commenced:	December 1, 2019
Completed:	July 27, 2020

Abstract

Named entity recognition (NER) is a challenging task, especially for morphologically rich languages like German. The complex language structures and intricate compound nouns, create difficulties in the localization of named entities. The process of recognizing and labeling named entities in an unstructured text can be attempted with many approaches but benefits most from deep learning methods. Current state-of-the-art approaches describe two broad ways for NER applications using language models: Fine-tuning and feature-based. The right choice of approach, suitable neural network architecture, and feature representations influence the performance of the system and are the key to success. The extensive research in this thesis investigates current achievements and methods in the sequence labeling task, NER, and point out difficulties that arise from the German language. The evaluation focuses on the two broad ways for NER and performs a comparative evaluation of the fine-tuning approach from BERT and feature-based approach from Flair on their performance on three German datasets.

This thesis presents methods of deep learning, which form the basis for the approaches compared in the evaluation. Different configuration settings of the approaches are evaluated on the CoNLL-03, GermEval and a complaint dataset. Besides, the performance of the models is examined in a cross-corpus experiment. The evaluation presents overall good results for both approaches but slightly better results with the feature-based approach from Flair. The experiments also reveal new best performance with the feature-based approach (86.62 F_1-score) on the GermEval-14 dataset. In addition, experiments on a small domain-specific dataset show that even with a low number of entities, good results are possible.

Kurzfassung

Named-entity recognition (NER) oder Eigennamenerkennung ist eine anspruchsvolle Aufgabe, insbesondere für morphologisch komplexe Sprachen wie Deutsch. Die komplexen Sprachstrukturen und komplizierten zusammengesetzten Wörter wecken Schwierigkeiten bei der Lokalisierung von Entitäten. Die Erkennung und Klassifizierung von Entitäten in einem unstrukturierten Text kann mit vielen Ansätzen versucht werden, profitiert jedoch am meisten von Deep-Learning-Methoden. State-of-the-art Ansätze beschreiben zwei Möglichkeiten für NER-Anwendungen unter Verwendung von Sprachmodellen: Fine-tuning (Feinabstimmung) und feature-based (Merkmalansatz). Die richtige Wahl des Ansatzes, die geeignete Architektur des neuronalen Netzwerks und die Darstellung von Merkmalen beeinflussen die Leistung des Systems und sind der Schlüssel zum Erfolg. Die umfassende Forschung in dieser Arbeit untersucht aktuelle Errungenschaften und Methoden in der Aufgabe der Eigennamenerkennung und weist auf Schwierigkeiten hin, die sich aus der deutschen Sprache ergeben. Die Evaluierung konzentriert sich auf die beiden Ansätze für NER und führt eine vergleichende Auswertung des Feinabstimmungansatzes von BERT und des merkmalsbasierten Ansatzes von Flair durch. Die Leistung der Ansätze wird auf drei deutsche Datensätze ausgewertet.

In dieser Arbeit werden Methoden des Deep Learning vorgestellt, welche die Grundlage für die in der Evaluierung verglichenen Ansätze bilden. Verschiedene Konfigurationseinstellungen der Ansätze werden auf den Datensätzen CoNLL-03, GermEval und einem Beschwerdedatensatz ausgewertet. Außerdem wird die Leistung der Modelle untersucht, wenn das Training und das Testen mit verschiedenen Datensätzen stattfindet. Die Auswertung zeigt insgesamt gute Ergebnisse für beide Ansätze, wobei bessere Ergebnisse mit dem merkmalsbasierten Ansatz von Flair erzielt werden. Außerdem zeigen die Experimente eine neue Bestleistung mit dem merkmalsbasierten Ansatz (86,62 F1-Score) für den GermEval-14-Datensatz. Darüber hinaus zeigen Experimente mit einem kleinen domänenspezifischen Datensatz, dass auch bei einer geringen Anzahl von Entitäten gute Ergebnisse möglich sind.

Contents

List of Figures

List of Tables

Acronyms

ANN artificial neural network. 25

BiLSTM bidirectional long short-term memory. 17, 19, 23, 28, 34, 35, 36, 41

CBOW continuous bag-of-words. 17, 22

CNN convolutional neural network. 23, 25

CRF conditional random field. 17, 18, 19, 23, 33, 35, 36, 41

DNN deep neural network. 25

FFNN feedforward neural network. 25

GRU gated recurrent unit. 27

LSTM long short-term memory. 17, 27

MLP multilayer perceptron. 25

MSE Mean Squared Error. 26

NER named entity recognition. 3, 11, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 27, 28, 29, 31, 33, 34, 35, 36, 39, 40, 41, 45, 47, 48

NLP natural language processing. 15, 18, 19, 21, 23, 24, 25, 27, 29, 31, 32, 34

OOV out-of-vocabulary. 17, 22, 23, 24, 34, 47

POS Part-of-speech. 15, 23, 31, 34

RNN recurrent neural network. 17

SGD stochastic gradient descent. 26, 34

SVM support-vector machine. 18

1 Introduction

In recent years, major steps for tasks in the research area of natural language processing (NLP) were taken. Due to the increasing computing capacities, new model architectures and pre-trained language models emerged. Thus, NLP tasks like named entity recognition (NER) receive greater attention and, above all, achieve high-quality results. NER is a well known sequence labeling task for identifying named entities in a sentence. For example, in the case of "Larry Page is one of the co-founders of Google." pre-defined classes, appropriate for this entity, Person: [Larry Page] and Company: [Google], could be extracted. Since entities contain semantic clues, the identification of entities is crucial for different applications of the NLP family, like automatic summarization, automated question answering, or sentiment analysis.

The methods for recognizing named entities went through several phases of change. Rule-based techniques that utilize hand-crafted rules, dictionaries, and syntactic-lexical patterns were common in the first efforts [HGA+98] [KH98] [BRM98]. With the rise of computational power and machine learning algorithms, unsupervised [NTM06] and supervised systems [YCL+18] [CN02] [LMP01] emerged and formed a good foundation for NER. The desire for a deeper understanding of the language led to another promising candidate, namely deep learning. As an area of machine learning, deep learning enables computational models to learn representations and discover the complex structure of large datasets [LBH15]. Because of the great success of deep learning methods in the recent years, the idea has shifted from task-specific feature-engineering to end-to-end systems with profound representation and contextual understanding of the underlying language. These representations, also known as word embedding, are a crucial component in such approaches. Word embeddings map words or phrases into an n-dimensional vector space, with the purpose of quantifying and categorizing linguistic features. Deep learning architectures usually learn word embeddings by training over a large collection of raw textual data. Taking advantage of novel architectures, exploiting nearly unlimited available data and the availability of pre-trained language models are one of the reasons why deep learning methods are becoming common practice in current NER applications.

Several NER approaches exploit deep learning methods, for example BERT developed by Devlin et al. [DCLT19], which provides state-of-the-art performance or the framework Flair developed by the Humboldt University of Berlin with Akbik et al. [ABV18]. Although they are similar in their performance, they differ from each other in terms of architectural design and approach. BERT offers a transformer-based approach, in which a language model is pre-trained on an enormous amount of raw data and then fine-tuned on a downstream task such as NER or POS tagging. Whereas Flair proposes a novel character-level language model that intents to extract contextualized word embeddings from sequences of characters. Still, they share the same ambition of deeply contextualized word embeddings, to catch latent semantic or syntactic clues. Even though both systems achieve high results for numerous downstream tasks in English, they perform significantly worse on morphologically complex languages such as German. In most Romance languages such as English or Italian, an advantageous aspect is that mainly proper nouns are capitalized, which can be utilized as a feature. Sequence labeling for German turns out to be challenging, as not only proper names are capitalized, but also nouns in general. Additionally, a compound including proper names, e.g.: “Donaudampfschiffahrtsgesellschaft” (steamboats company on the Danube), is not uncommon and holds extra challenges for the identification of entities. Given the hurdles of the German morphological complexity and the likely superiority of deep learning methods, a comprehensive evaluation of current state-of-the-art approaches allows an insight and determines the potential for a German NER application.

The aim of this thesis is to examine current state-of-the-art deep learning methods with a focus on the pre-trained models from BERT and Flair. The contribution is primarily an in-depth evaluation and a comparison of BERT and Flair achievements on different German NER tasks. The evaluation is carried out with three different datasets: CoNNL-03, GermEval-14, and a complaint dataset.

Outline: Chapter 2 presents related work in the field of NER-related deep learning methods especially for German NER. Different types of embeddings and deep neural networks are covered in Chapter 3. A detailed explanation of the components of BERT and Flair, followed by the experimental setup, is presented in Chapter 4. Chapter 5 presents and discusses the results of the evaluation. The final Chapter 6 ends the thesis with a summary and suggestions for future work.

2 Related Work

The following Section 2.1 showcases several approaches and contributions in word embeddings and deep neuronal networks that strengthened the motivation for deep learning methods in NER. Furthermore, Section 2.2 presents previous and recent achievements in German NER.

2.1 Deep learning methods for Named Entity Recognition

Word embeddings, a crucial unit of deep learning models, has been responsible for many state-of-the-art results in NER applications [DCLT19] [ABV18] [PNI+18]. Modern word embeddings, part of the group of models called Word2Vec, were introduced by the authors Mikolov et al. [MSC+13] [MCCD13] as continuous bag-of-words (CBOW) and Skip-gram. While CBOW computes word embeddings by predicting the word based on his surrounding context, Skip-gram does the opposite and tries to predict the surrounding words given the current input. Although both systems are not based on deep neuronal networks, their output can be processed by them. Improving the performance of an NER system with the word embeddings from a Word2Vec model is explored by the work of Siencˇnik [Sie15] as well by Joshi et al. [JHVR15]. In 2014, another word representation approach was presented, namely GloVe by Pennington et al. [PSM14]. One of the disadvantages of Word2Vec models is that they consider only a small window of context and don't deal with the vast amount of co-occurrences in the corpus. GloVe utilizes repetition in the data and therefore considers global and local statistics. The authors showed better results on the NER task than with Word2Vec models. A disadvantage of the previously mentioned approaches, failure to handle out-of-vocabulary (OOV) words, led to another approach from Bojanowski et al. [BGJM17], called fastText. Introduced as the concept of word embeddings at subword-level, word embeddings are learned through the information of the whole word and its subwords. Because of the subword information, a fastText model can handle OOV words better than Word2Vec models where OOV words are assigned random values.

A fundamental problem of the presented concepts is the disregard of ambiguities and the context-dependent nature of words. Peters et al. [PNI+18] introduced a novel type of deep contextualized word representations. Their model ELMo trains a multi-layer, bidirectional, long short-term memory (LSTM) language model and extracts the hidden state of all the internal layers to obtain the word embeddings. By allowing the model to learn a weighted average of all layers, they achieve state-of-the-art performance. The current state-of-the-art models from Flair [ABV18], ELMo, and more [MBXS18] [CLML18] use the recurrent neural network (RNN) architecture in their applications. Additional research combines the BiLSTM architecture with a conditional random field (CRF) layer at the top, to efficiently use past assigned entity categories when predicting the current entity [MH16] [PA18]. In 2018, Akbik et al. [ABV18] proposed a novel character-level word representation. By modeling the words and phrases as sequences of characters, the word embeddings are varying based on the context and operate better with OOV words. A different approach to RNN-based systems build the Transformer, first introduced in the research by Vaswani et al. [VSP+]. Google, under the supervision of Devlin et al. [DCLT19], proposed with BERT a transformer-based system to pre-train a language model. Unlike OpenAI-GPT [RWC+] where Radford et al. [RWC+] pre-trained a shallow left-to-right and right-to-left transformer-based language model, BERT is trained with a "masked language model" and "next sentence prediction" objective. Devlin et al. [DCLT19] confirm state-of-the-art performance through the deep bidirectional character of BERT.

2.2 Approaches to German Named Entity Recognition

In the earliest attempts to identify and classify named entities, only little research was devoted to the German language. One of the first attempts, namely PRONTO [Lan01], utilized different heuristics like grammar rules and common patterns in names and surnames for the entity recognition of person names. The research from Piskorski and Neumann [PN00] likewise applied compound and tagging rules for recognizing organization, person, and location names. Furthermore, they exploited lexicons like gazetteers and name lists to increase the recall. Volk and Clematide [VC01] presented several rules in their rule-based approach for the recognition of person, geographical, and organization entities. They concluded that German NER profits from extensive grammatical rules and lists but also mentioned the variety in length and structure of German words complicates the recognition. Initial approaches with machine learning algorithms were developed by Florian et al. [FIJZ03]. Their evaluation, conducted with several classifiers, presented a good result for German with a Robust Risk Minimization Classifier. Just like researches before them, they mentioned problems for German NER because of the morphological complexity and in addition a larger number of unique words in the corpora than in English corpora. Rössler [Rös04b] proposed an approach based on a linear support-vector machine (SVM) in which the automatic creation of reliable domain-sensitive lists of named entity is addressed. With a reasonable amount of labeled data and a large number of unlabeled data, Rössler achieved competitive results on the extraction of person names on the CoNNL-2003 dataset [Rös04b]. Additionally, Rössler adapted the aforementioned system to a biomedical domain [Rös04a].

The approach from Qi et al. [QKC+09] investigated feature distribution of words as an extra feature in the supervised training of a classifier. They applied their approach on several NLP tasks and achieved good results in German NER. Faruqui and Padó [FP10] presented the utilization of distributional similarity features learned over a large unlabeled corpus. They reported good results for the CoNLL-2003 Shared NER task when features are trained from the corpus in the same genre. The research of Benikova et al. [BMPB15] presents a model based on a CRF [LMP01], predicting the most likely label to an input. According to the paper their best configuration achieved an F_1-Score between 76 and 78 on the GermEval-2014 dataset, however they included only four out of 16 named entity classes of the dataset. Another CRF approach is presented by Hänig et al. [HBT] where the authors came to the conclusion that the CRF classifier produces better results than maximum entropy classifiers. A modification of an English and Russian rule-based NER approach to German was analyzed by Druzhkina et al. [DLS16]. They stated that several changes to the system have to be made because of the complexity of organization names, composites and foreign words in the German language.

Modern deep neural architectures were applied by the work of Akhundov et al. [ATG18]. Their sequence labeling approach is based on CRF on top of a BiLSTM. The authors reported high performance in the combination of word and byte embeddings with the prediction from the CRF layer. Riedl and Padó [RP18] investigated the effect of different big- and small-data situations on BiLSTMs and CRFs. In their evaluation, they didn't consider only the CoNNL-2003 and GermEval-14 but also German historical datasets. Comparable to Akhundov et al. [ATG18], Riedl and Padó stated a combination of BiLSTM and a CRF layer yields the best performance, though they struggle with short datasets. One novel approach for German NER was proposed by Akbik et al. [ABV18]. Their system is based on a bidirectional character-level language model that provides contextual word embeddings. By capturing deep linguistic features taking into account the context, the authors were able to achieve a state-of-the-art performance with their model for German NLP tasks and in several other languages. 2018 introduced Devlin et al. [DCLT19] BERT, a different type of system suitable for a German NER application. This approach pre-trains a transformer-based language model on a large amount of data. By the special pre-training tasks and characteristic of the Transformer, the word embeddings are deep contextualized and reason for state-of-the-art results. Their research only considered the performance on the English part of the CoNLL-2003 NER task. Nevertheless, a similar good result can be expected for the German language.

BERT and Flair are a suitable choice for NER applications. The state-of-the-art performance on the NER task motivates an in-depth investigation of deep learning methods and a comprehensive evaluation of BERT and Flair on NER for German. Because none of the work mentioned in this chapter conducted an evaluation of both approaches, this contribution aims to keep up the motivation for research and work in German NER

3 Word representations and Neural Networks

This chapter contains a closer look at concepts of word representations of words and different types of neural networks that are essential for the work in this thesis. The first three sections explain different mechanisms for feature representation. Word embeddings in Section 3.1, the benefits of character embeddings in Section 3.2 and details about contextual embeddings in Section 3.3. Insights about the structure of feedforward neural networks are provide by Section 3.4. The architecture and different forms of recurrent neural networks are demonstrated in Section 3.5. Lastly, Section 3.6 presents the Transformer architecture, a crucial component of BERT.

3.1 Word Embeddings

At the beginning of every current state-of-the-art deep learning method for NER, is the process of learning word embeddings of the input data. Textual data in its pure shape isn't processable for neural networks and therefore needs to be in a format on which a model can operate. An essential technique for representing words in a machine-friendly form are vectors embedded in a vector space. One-hot encoding (Figure 3.1) is a simple word embedding algorithm where each category of the input is a binary entry in a vector representation. One problem with such sparse representations is the rising number of dimensions with every additional category, also called the "curse of dimensionality" [Bel03]. Another problem of sparse representations, is the ability to capture and measuring similarities between words. Word representations map words to low-dimensional vectors with real values where each dimension of the vector space describes a latent feature. When measuring the cosine of the angle of two words, a smaller angle indicates a higher similarity. This concept is described as cosine similarity and can be used to determine whether two words are similar or not. Considering the simplified example in Figure 3.2, even more advanced calculations are feasible.

Although the calculation produces not exactly the same vector, the concept of capitals is captured. Through many epochs of learning with large datasets, word embeddings capture linguistic properties of words. Common natural language processing (NLP) tools to represent words as vectors belong to

$$\text{The apple tastes sweet} \rightarrow \begin{array}{rl} \text{The} & [1,0,0,0] \\ \text{apple} & [0,1,0,0] \\ \text{tastes} & [0,0,1,0] \\ \text{sweet} & [0,0,0,1] \end{array}$$

Figure 3.1: One-hot encoding of each category (word) in the sentence. All entries in a one-hot vector consists of 0s except of a single 1, which is used to identify the word.

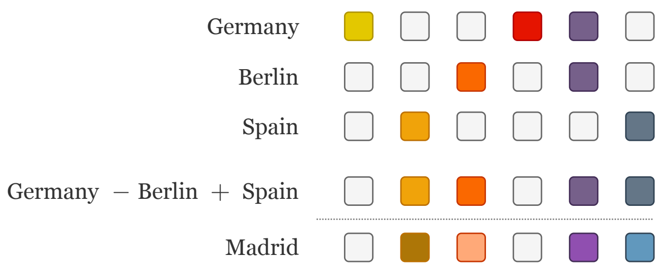

Figure 3.2: Vector illustration of word embeddings. Vectors simplified with colors. The vector calculation produces a word vector that is close to that from Madrid.
Another famous example is $King - Man + Women = Queen$.

the class of Word2Vec models, like CBOW [MSC+13] and Skip-gram [MCCD13]. While a CBOW model is trained to predict a word based on the surrounding words, a Skip-gram model is trained to predict the surrounding based on a word.

Another popular model is GloVe [PSM14], shorthand for Global Vectors. Just like Word2Vec models, GloVe models incorporate the surrounding context when processing a word but also take word co-occurrences into account. An extension to Word2Vec models are word embeddings from fastText models [BGJM17]. Instead of learning word embeddings only for the full-word, fastText also utilizes the structure of words. FastText performs this by producing n-grams of characters for each word in the input. Given the term "Vector" and $n = 3$, fastText models learn word embeddings through full-word "Vector" and 3-grams $<Ve, Vec, ect, cto, tor, or>$[1]. FastText models operate well with unseen words because they treat words as n-grams of characters and therefore most likely have learned representations of parts of that unseen word. On the contrary, standard Word2Vec and GloVe models can only handle words that were present in the training data so they fail when facing unseen or OOV words. Despite Word2Vec and GloVe models seeming to consider the context, they are referred as "context-free" models. Because words that occur in the same context, especially polysemous words, have the same embeddings regardless of the context. A polysemous word is a word with different meanings depending on the context. In both sentences "A mouse is a popular pet variety" and "I prefer a black computer mouse" the term "mouse" would get the same word embedding from a Word2Vec or GloVe model.

However, the advantage of fastText is quite practical for German text. The notorious long words are usually OOV words and therefore make it difficult to accurately identify named entities. The method of fastText handle this by breaking down long words into a simpler fashion, where its more accessible to produce word embeddings. This is one of the reasons why fastText is part of current state-of-the-art applications and provides good achievements for German NER [ABV18] [DCLT19].

[1] "<" describe the beginning and ">" the ending of a word

3.2 Character Embeddings

Word embeddings capture linguistic features by working on full-word information or in the case of fastText even on sub-word information. Character Embeddings on the other hand incorporate character-level information and therefore treat words as sequences of characters. This method has many benefits for NER applications especially for morphologically rich languages such as German. The performance of German NER in comparison to English NER is considerably poorer due to the complex language structure. Character Embeddings treat words as sequences of characters and therefore are able to produce reliable word representations for OOV words and infrequent words. Furthermore embeddings from character-level models work well with misspelled words as they learn how the word is structured. Lample et al. [LBS+16] proposed an approach where they combine character-level and word representations. Their neural network is composed of a CRF layer based on an RNN architecture. The authors mentioned that domain-specific sequence labeling tasks benefit from character-level features. In addition, they are useful for grammatically complex languages and deal favorable with OOV words. Character embeddings are not only beneficial for NER but also for other sequence labeling tasks such as Part-of-speech (POS). Similar to NER, the goal of POS tagging is labeling words with a corresponding part of speech (noun, verbs, etc.). The work from Labeau et al. [LLA15] employs a convolutional neural network (CNN) with an RNN-based architecture to extract character-level features. Evaluated on German text data, their high results showed that character-level features address the OOV words and produce superior word embeddings for unique and complex words.

The quality of the word and character representations depends on how well the neural network processes the surrounding context of a word. Classic word-level and character-level representation produce useful vector representations but aren't deeply context-sensitive. A novel type of word representation, called contextual word embeddings, leverages the context and therefore assimilates meaningful linguistic clues.

3.3 Contextual Embeddings

Recent successes in the large family of NLP are, among other things, traceable to the concept of deep contextualized word embeddings. These representations capture the complex characteristics of words and how the characteristics vary different contexts. Deep neural networks like RNN and Transformers produce word embeddings through their internal states as a function of the surrounding context. Considering the example in Section 3.1, contextualized word embeddings would now produce different word vectors because of the distinct context. Many NLP tasks profit from this method, especially sequence labeling task like NER where the context has an active impact on whether a word corresponds to an entity class or not.

Peters et al. [PNI+18] introduced with ELMo (Embeddings from Language Models) a deep contextualized word representation. Contextualized in the sense that word representations are learned through considering the whole sentence before assigning a word vector. ELMo representations are therefore deep contextualized as a result of the function of all internal states within the bidirectional language model. Further, the language model incorporates character-level features and allows the system to produce robust representations notably for OOV words. ELMo's architecture is based on bidirectional long short-term memory (BiLSTM) a type of RNN. With a corpus of textual data, the

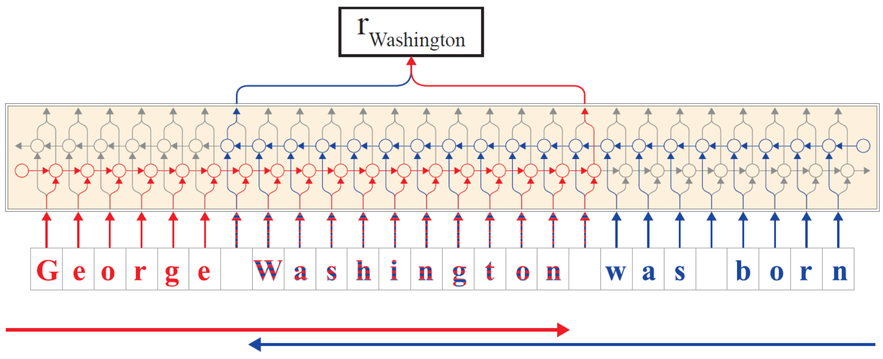

Figure 3.3: Visualization of extracting the contextual word embedding for “Washington” with Flair’s character-level language model. To form the final word embedding the hidden states of the forward language model (red) and backward language model (blue) are concatenated. (Source: Akbik et al. [ABV18])

language modeling task consists of forecasting the next word in a sequence of words (sentence). In particular, two language models are trained, the forward model computes the probability of the past tokens while the backward model does the same but given the future words. ELMo now obtains the contextualized word representations by concatenating the hidden states of both language models and a weighted summation based on the target NLP Task, e.g. NER. During training on the end-task ELMo learns the weights thus makes the word embedding task-specific.

Contextual Embeddings likewise benefit from the idea of modeling words and the context as sequences of characters. Akbik et al. [ABV18] proposed a type of representation that combines the character-level features and the context of the textual data and thus refers them as contextual string embeddings or Flair embeddings. Similar to ELMo embeddings, Flair embeddings capture word meaning in context and therefore produces different representations for ambiguous words. Because the language model exploits character-level features, the gained embeddings make it easier to handle OOV words and model linguistic features like prefixes and endings. Figure 3.3 visualizes the process of how the authors extract the contextualized word embeddings for the term “Washington”. The main idea behind the approach is the pre-training of two character-level language models, one forward and one backward. The forward language model captures semantic-syntactic information from beginning of the sentence to after the last character of “Washington”. The backward language model operates in a reverse fashion, from the end of the sentence to after the first character of “Washington”. Thus the output hidden states contain information propagated to the respective point. The final embeddings are then formed through concatenation of both output hidden states. Detailed information on the use of Flair embeddings in downstream tasks like NER is explained in Section 4.2.

With BERT (Bidirectional Encoder Representations from Transformers) [DCLT19] a new method for training language models and the resulting contextualized word embeddings, was introduced. Different to ELMo and Flair, BERT’s word embeddings originate from a novel type of language model, called Transformer [VSP+]. The contextual word embeddings generated by BERT are the result of a “masked language model” task and “next sentence prediction” task. Through these procedures, the model is forced to keep contextual representations for every token and therefore produces contextual word embeddings.

Details on the structure and characteristics of Transformers can be found in Section 3.6. Detailed insights about pre-training language models with BERT and how the contextual word embeddings are utilized in NER, are mentioned in Section 4.1

3.4 Feedforward Neural Network

Core of modern natural language processing (NLP) approaches and one of the reasons for state-of-the-art performance are deep neural networks (DNN), a special type of artificial neural networks (ANN). A simple basis for the large family of DNNs are feedforward neural networks (FFNN). The concept behind this network is the forward propagation of information from the input layer, through the units of the hidden layer down to the output layer, visualized in Figure 3.4. Different types of FFNNs propagate information only in forward direction, such as convolutional neural networks (CNNs) or Auto-Encoders. This section mainly focus on the basic kind of FFNN, i.e. multilayer perceptron (MLP).

An MLP is an extension to the antiquated single-layer perceptron. The term perceptron describes an artificial neuron and unit of a neural network which takes separately weighted inputs, adds them up to an activation signal and applies an activation function such as the unit step function. This is one of the most basic activation functions because the result can only be one of two possible values. Neurons with such a binary function are used for classification tasks and called “True perceptrons”. A single-layer perceptron is a basic network and performs only binary classification, which is a disadvantage for data that cannot be separated linearly. An MLP consists of several neurons (hidden units), except for the input nodes, that use non-linear activation function. Since the neurons in an MLP can utilize arbitrary activation functions, the neurons are unrestricted to perform either classification or regression. In a strict sense, the neurons of an MLP are not perceptrons because of the arbitrary selection of the activation function. There are several kinds of activation functions which are utilized by the hidden units. Step functions like Heaviside (3.1), linear functions but also complex functions such as sigmoid (3.2) or tanh (3.3). The number of neurons per layer as well as the count of hidden layers is not precisely defined and depends on the task, dataset and desired performance.

$$f(x) = \begin{cases} 1, & \text{if } w * x + b > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

$$\varphi_{\text{logistic}}(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

$$\varphi_{\text{tanh}}(x) = \tanh(x) \tag{3.3}$$

FFNNs are mostly used for supervised tasks where it is already known what the network has to achieve. Training an FFNN therefore takes place in a supervised fashion. Over many epochs input and desired output is fed into the network, so that the system captures the relations between input and desired output. To evaluate the performance during the training a cost function is defined. The cost function is always single-valued and demonstrates the difference between approximation of the

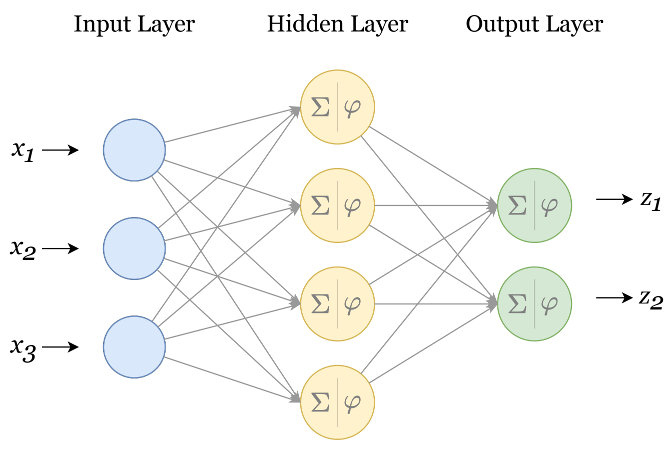

Figure 3.4: Example of a simple feed forward neural network. With one hidden layer consisting of four hidden units (neurons). The inputs x are summed up and form the activation signal which will be forwarded to the activation function φ. Finally the output layer produces the outputs z.

network and the desired output. A common and simple cost function is the Mean Squared Error (MSE) also be called quadratic cost:

$$MSE \equiv \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.4}$$

Where N describes the total number of inputs, y the desired output and $\hat{y}$ the output of the network. There are also several other cost functions like cross entropy, exponential or Hellinger distance. The choice of an adequate cost function depends on the data and the objective to be achieved. In order to minimize the cost function and thus improve the approximation of the network, gradient based learning is used. The algorithm for training machine learning models, gradient descent, is an iterative method and can also be utilized for training DNNs. The basic idea behind gradient descent is to make small changes to the models parameters (weights and biases) that lead to the global minimum in small steps. Over many epochs of training this will continually minimize the cost function.

Training with a large amount of data is associated with certain difficulties. The computing time for a large number of training samples and the adjustments of potentially millions of weights and biases are examples for this. An intention to speed up the iterations is stochastic gradient descent (SGD). Instead of calculating the gradient from the entire dataset, the estimation from a randomly selected subset is chosen. Backpropagation, another algorithm for training a network, calculates the gradient of the cost function with respect to the weights of the network. This algorithm enables the simultaneous adjustment of the weights, based on their impact, in one forward and backward pass of input. Combining these two methods is a common approach in training a deep neural network (DNN).

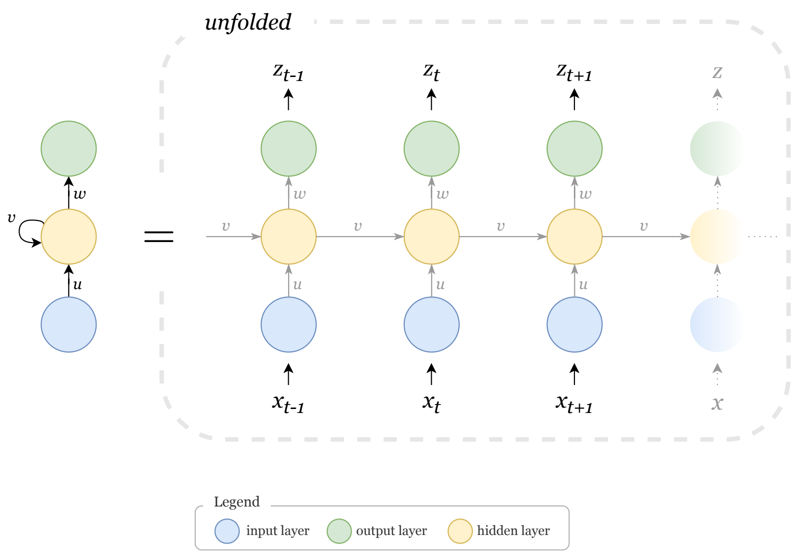

Figure 3.5: Recurrent neural network with three layers. The parameters u, v, w are shared across all time steps and if training takes place, get adjusted.

3.5 Recurrent Neural Network

In the previous section the structure and concept behind FFNNs are described. However, basic neural networks assume that all inputs and outputs are independent of each other. But in the field of natural language processing (NLP) the data is in the form of sequences and especially the context depends on the past and future. The architecture of a recurrent neural network (RNN) is meant to make use of the sequential information. The idea behind RNNs is to link information from previous states to the current and use this “knowledge” to predict a desired outcome. Their internal states or hidden states operate as a kind of “memory” that captures information from previous states. This approach is useful for language modeling and in NLP applications such as named entity recognition (NER), speech recognition, machine translation and others. The basic structure of an RNN is shown in Figure 3.5. This figure displays the recurrent nature of the network since each output depends on the previous state output.
When training an RNN or an FFNN with gradient based algorithms and backpropagation, especially those with many layers, two notorious problems will arise: Vanishing or exploding gradient [Hoc] [BSF94] [PMB13]. Vanishing gradient is described as the problem of gradients getting close to 0 and therefore prevent an update of the neural network’s weights. In the worst case the training stop completely. This happens with a rising number of layers in the network but also by use of certain activation functions such as tanh (3.3) . When large gradients occur during training, this is considered as a exploding gradient problem. This problem is manifested by very large changes in the weights of the neural network, making the network unstable. To work around the vanishing gradient problem, other activation functions can be used, e.g. rectifier (ReLU) [HSM+00]. Another solution and common application are long short-term memorys (LSTM) or gated recurrent units (GRU). Proposed by Hochreiter and Schmidhuber [HS97] are LSTMs, a special kind of RNN architecture, suited to avoid the vanishing gradient problem. As shown in Figure 3.5 a basic RNN has only a single hidden layer whereas the units of an LSTM are composed of a cell, an input gate, an output gate and a forget gate (Figure 3.6). The first part builds the forget gate. Previous hidden state and

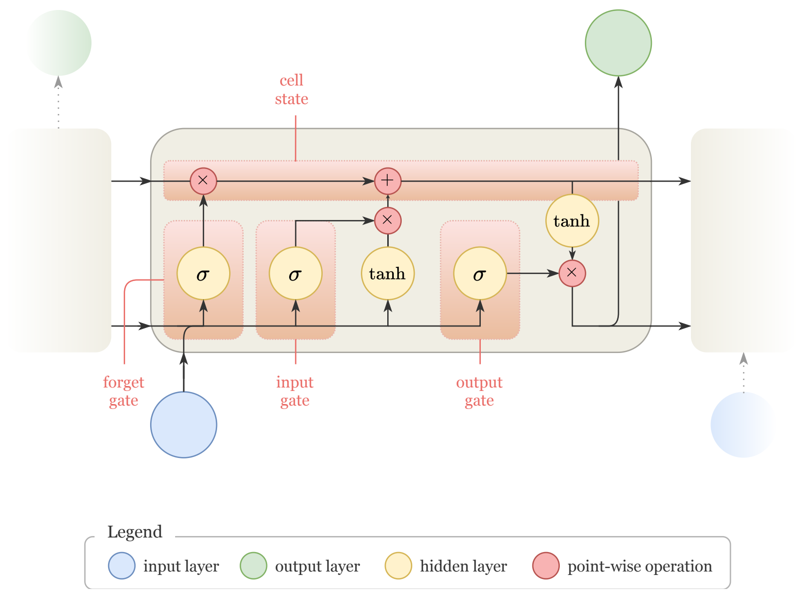

Figure 3.6: Long short term memory unit with sigmoid (σ) 3.2 or tanh 3.3 activation function in the hidden layer. The upper horizontal line is called the cell state, information flows over it. The "gate layers" are able to add and remove information to the cell state.

current input is passed to a sigmoid function, which then decides what information should be thrown away or kept. Next the input gate forms the additional input for the cell state. The upper horizontal line, called cell state, serves as "highway". As information flows on this "highway", gates are able to append information to the cell state or the information pass unchanged. With this approach, an LSTM unit allows the gradient to flow unchanged and therefore avoiding the vanishing gradient problem. Finally the output gate computes from the cell state, previous hidden state and current input, the new hidden state. Cell state and new hidden state is then passed to the next time step.

Proposed by Cho et al. [CMG+14] is the gated recurrent unit (GRU) a newer variation of an LSTM, although they are designed in a similar fashion. The GRU is composed of an update gate and a reset gate. GRU's update gate is responsible for adding and removing new information, therefore combining the process of the forget gate and input gate of an LSTM. The reset gate decides what information to "forgotten" from the previous hidden state. Just like an LSTM, a GRU eliminates the vanishing gradient problem by storing and distilling information through the gates.

Basic RNNs, LSTMs and GRUs process a sequence of tokens from past to the future. To establish a greater contextualized understanding of the underlying language, it is advisable to assemble two networks [STN19]. One processing the token from past to the future and one processing from future to the past. These networks are referred to as "bidirectional" networks, hence BiLSTM. It has been shown that BiLSTM-based models perform better on prediction than regular LSTM-based models [STN19]. This is one reason why bidirectional network architectures are part of many NER applications, for instance ELMo and Flair.

3.6 Transformer

The Transformer was first introduced in the research “Attention is all you need” by Vaswani et al. [VSP+]. Prior to the introduction of the transformer architecture, many natural language processing (NLP) applications were based on recurrent neural network (RNN) architectures some with extra attention mechanisms. The attention mechanism approach was proposed by Bahdanau et al. [BCB16] and it has emerged that it boost the quality of NLP applications like machine translation [LPM15], named entity recognition (NER) [NDN19] and others [CBS+15]. The concept behind this mechanism is that it allows a model to determine relevant parts of a sequence when processing an element of that sequence. The attention mechanism can be considered as encoding a token by the weighted sum of its context.

The two main components of the Transformer are a set of chained encoders and a set of decoders. As pictured in Figure 3.7, the first step of the Transformer encoder is adding positional information to each word embedding. The idea here is that additional positional information provides meaningful distances between the word embeddings. The encoder then passes each input vector into a self-attention layer. As aforementioned, this allows the attention mechanism to find information about the elements which refer to other elements in the sequence. To achieve this, each input element is weighted according to its relevance and used as soon as the output is computed. This method is comparable to the mechanism in an RNN where the hidden states provide relevant information about the elements. In the next step the output of the self-attention layer and tokens from a residual connection (dashed line) are passed to the “normalize” layer. This procedure is common for training neural networks and reduces training time significantly [BKH16]. Once the vectors are normalized, each individual element is entered into a position-wise feedforward neural network (FFNN). Finally the vectors are normalized a second time and will be passed to next encoder block of the transformer. The authors Vaswani et al. [VSP+] stack six Transformer encoders and decoders, whereas in BERT they use settings with 12 and even 24 Transformer blocks of encoders [DCLT19]. The decoder architecture is rather similar to that of an encoder. Like the encoder, the decoder’s sub-layer consist of a “self-attention” layer, “normalize” layer and an FFNN but in between there is an additional layer, called “encoder-decoder attention” layer. This extra layer supports the decoder to focus on the relevant parts of the input sentence and is important for a machine translation task.

Due to the fact that each element of the input sequence follows its own path in the encoder, the processing of an element is not coupled with the previous one, like in an RNN. Therefore a transformer-based model allows significantly more parallelization and training on much more data [VSP+]. This advantage is one of the reasons for the development of pre-trained transformer-based language models like OpenAI’s GPT and BERT.

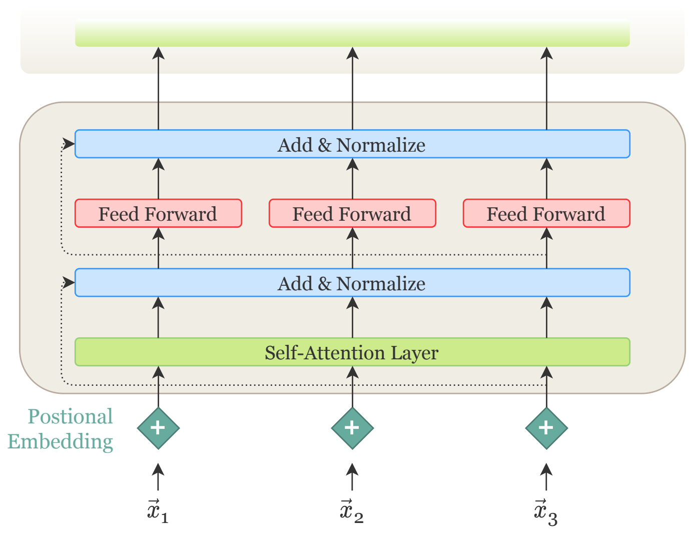

Figure 3.7: Bottom encoder block of the Transformer. Each word vector x_t passes through the encoder on its own and is ultimately entered into the succeeding encoder block.

4 Fine-tuning and Feature-based Approach

The aforementioned methods build the base for modern named entity recognition (NER) applications and led to two major strategies: Fine-tuning and feature-based, when facing the NER task. In fine-tuning approaches, like BERT and OpenAI's GPT [1], the model's pre-trained parameters are slightly modified or fine-tuned during training on the downstream task, such as named entity recognition (NER). Whereas in feature-based approaches like Flair the pre-trained parameters of the language model are frozen and instead an attached sequence labeling model is trained over many epochs on the downstream task. The following sections provide insights into the process of pre-training the language models and further steps for the NER task. Lastly, Section 4.3 describes the experimental setup for the comparative evaluation.

4.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) was first introduced in 2018 and developed by Devlin et al. [DCLT19]. BERT's model architecture consists of a multi-layer bidirectional Transformer encoder, based on the implementation of Vaswani et al. [VSP+]. The pre-trained BERT models provoked a paradigm shift in the field of natural language processing (NLP). Because of the transformer-based architecture, BERT is primed to pre-train representations from a vast amount of unlabeled text and captures contextual clues of the underlying language. Therefore, the pre-trained BERT model builds the base and can be fine-tuned with one additional output layer during supervised training on the NER task. The availability of pre-trained models is one of the reasons for the paradigm shift and the success of BERT models. Developers and research in the field of NLP are freed from the process of collecting a vast amount of unlabeled textual data for pre-training a language model. A task-specific dataset and an appropriate output layer are sufficient. Another benefit of BERT is the customizability for different types of NLP tasks, such as NER, POS tagging, question answering (QA), natural language interference (NLI) and more.

4.1.1 Pre-training of BERT

As mentioned in Section 3.5, deep bidirectional models are strictly more powerful [STN19]. "Bidirectional" language models like ELMo, according to Devlin et al. [DCLT19], are not deeply bidirectional. One reason is the independent training process of two separate language models and their "shallow concatenation". A limitation of other transformer-based models like OpenAI GPT is their unidirectional nature. Unidirectional, because they process word sequences only in one direction. To enable a deep bidirectional pre-training the authors proposed two different language

[1] https://openai.com/blog/better-language-models/

modeling tasks that grant this intention. The first is called “masked language model” (MLM). In this pre-training task, 15% of the tokens in a sequence are selected randomly as masked tokens and have to be predicted. To force the model to employ the contextual surrounding, the selected tokens are replaced with a special token: `[MASK]`. A drawback the authors mentioned, is the emerging mismatch between pre-training and fine-tuning, because the `[MASK]` token is not encountered during fine-tuning. The following strategy overcomes this problem:

- In 80% of the cases, replace the token with `[MASK]`.
- In 10% of the cases, replace with a random token.
- In 10% of the cases, keep the token unchanged.

With this approach the model has no information which tokens it should predict or which token have been replaced, hence the model is compelled to keep contextual representations of every token.

The second language modeling task is called “next sentence prediction”. As the name suggests, this task forces the model to learn relationships between sentences. This procedure is useful for NLP tasks such as question answering (QA) and natural language inference (NLI). A pre-training examples for the “next sentence predection” task is illustrated below:

- Input = `[CLS]` the `[MASK]` tastes sweet `[SEP]` it `[MASK]` fit nicely in a cake `[SEP]`
  Output = `IsNextSentence`
- Input = `[CLS]` the apple `[MASK]` sweet `[SEP]` a group of owls is `[MASK]` a parliament `[SEP]`
  Output = `NotNextSentence`

Sentence pairs for pre-training on this binary classification task, are 50% actual successive sentences and 50% are non-successive. The special token `[CLS]` is a special symbol for classification, while `[SEP]` is inserted at the end of each sentence. In the paper, the authors presented two pre-trained models of BERT:

- BERT$_{BASE}$: $(L = 12, H = 768, A = 12, B = 256, DP = 0.1, LR = .0001)$
- BERT$_{LARGE}$: $(L = 24, H = 1024, A = 16, B = 256, DP = 0.1, LR = .0001)$

Where L is denoted as number of layers, H as hidden size or hidden units and A as the number of attention heads in each layer. B represents the batch size (sequences of tokens), DP or drop out is the likelihood that the outgoing edges of a neuron are set to zero. This technique prevents a model from overfitting. Learning rate (LR) is a parameter that controls how great the weights of the network are adjusted. The count of A is linked to an extension of the attention mechanism, which the authors take advantage of. “Multi-headed” attention is the key phrase. This mechanism expands the model’s ability to concentrate on relevant parts. Also, it provides the model multiple sets of “subspaces” to compute the attention with start conditions. The dataset for pre-training both models consists of English Wikipedia data with 2.500M words and an English book corpus with 800M words.

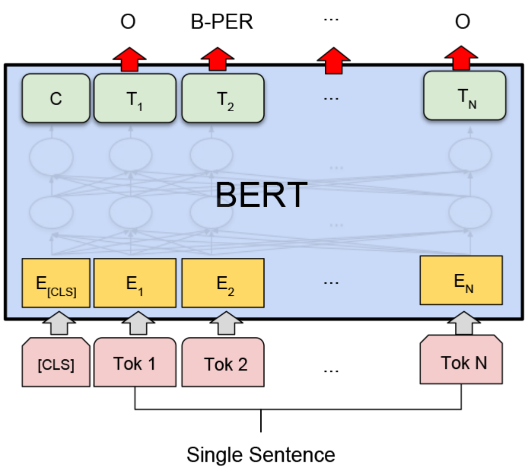

Figure 4.1: Illustration of BERT's structure for fine-tuning on the NER task. The structure is, apart from the output layer, the same for pre-training and fine-tuning. The pre-trained model's parameters can be utilized across different downstream tasks. (Source: Devlin et al. [DCLT19])

4.1.2 Fine-tuning approach for Named Entity Recognition

The process of fine-tuning a BERT model on a NER task follows the pre-training process. Through the two language modeling tasks, the model has learned contextual representations and is ready for the downstream task. Fine-tuning is performed on the task (e.g.NER) and therefore needs task-specific inputs, outputs, and an additional classification layer. Compared to other classifiers like RNN and CRF, the classifier used for BERT is more basic. The idea is to project the token hidden state to the tag set, with a subsequent normalization by a softmax function. A softmax function turns the "raw" logits scores into probabilities. The vector entry with the highest probability then defines the suitable tag. The Figure 4.1 illustrates the sequence labeling architecture of BERT. The input representations (yellow) are fed into the multi-layer Transformer encoder block. These representations are the sum of token embeddings, segment embeddings, and position embeddings. The token embeddings are WordPiece[2] [WSC+16] tokens. Segment embeddings take into account which sentence the token belongs to, while position embeddings track positional information of a token. The final output layer (green) predicts suitable tags for all tokens passed through the encoder.

The model's hyperparameters for pre-training and fine-tuning remain the same except for the batch size, learning rate and number of training epochs. The authors mentioned a batch size of 16 or 32, a learning rate out of this set: {5e-5, 3e-5, 2e-5} and a number of training epochs between 2 and 4 [DCLT19]. With these hyperparameters, the encoder and the classifier are jointly fine-tuned during training on NER.

[2]WordPiece is a word segmentation algorithm, for tokenization purpose

Not only fine-tuning but also a feature-based approach is realizable with BERT. In this scenario, a network extracts fixed features from the pre-trained model. The features then server as input to a sequence labeling model. This method provides computational benefits. Instead of pre-training expensive representations many times, it is advisable to compute them once and execute many experiments with cheaper models. Devlin et al. [DCLT19] reported results on a feature-based approach in their work. The contextual embeddings from BERT were used as input to a two-layer BiLSTM followed by the classifier. They examined several embeddings, extracted from different layers in BERT$_{BASE}$. BERT's feature-based performance on the English CoNLL-2003 NER task [TD03] is not far behind the fine-tuning approach. The best performance is achieved by BERT if the last four hidden layers are concatenated and used directly for the NER task. This demonstrates the versatile application possibilities of BERT, as the authors noted [DCLT19].

4.2 Flair

Flair is a state-of-the-art natural language processing (NLP) framework developed among others by the Humboldt University of Berlin. It emerged from the work "Contextual String Embeddings for Sequence Labeling" by Akbik et al. [ABV18]. The authors proposes a novel type of contextualized character-level word embeddings, called Flair embeddings. The embeddings are based on a character language model which is trained without any explicit concept of word and sentence structure. These representations capture linguistic features of the word based on the surrounding context and therefore provide different embeddings for polysemous words. In contrast to BERT embeddings, this approach models words and context as sequences of characters. Similar to fastText, there is an advantage when facing OOV or misspelled words and modeling subword structures like affixes. Experiments have shown that a feature-based approach with Flair embeddings outperforms previous approaches, especially for the German language, in tasks like named entity recognition (NER) and Part-of-speech (POS) tagging.

4.2.1 Pre-training of Flair

The pre-training process of the proposed character-level language model simpler compared to the pre-training process of a BERT model. The objective of a character-level language model is to learn a good joint probability distribution $P(x_{0:T})$ over sequences of characters x. While training with a large amount of textual data, the model learns $P(x_t|x_0, ..., x_{t-1})$. An estimation of the distribution over a character given the previous characters. The character-level language model is composed of a recurrent neural network (RNN) structure, more precisely utilizes long short-term memory (LSTM). Because the authors train two separate language models, one forward and one backward (Figure 3.3), the resulting language model can be considered as bidirectional. To optimize the cost function of the language models, stochastic gradient descent (SGD) is used as an optimizer. The following parameters have been selected for the proposed approach:

- Training LM: $(L = 1, H = 2048, B = 100, CG = 0.25, DP = 0.25, LR = 20)$

Where L refers to the one-layered LSTM and H represents the number of hidden states in the layer. B indicates the batch size, or numbers of examples from the overall sample. CG describes the gradients threshold to avoid exploding gradients. Dropout probability (DP) is the likelihood that the outgoing edges of a neuron are set to zero. That technique prevents a model from overfitting. Learning rate (LR) is a parameter that controls how great the weights of the network are adjusted. For the English model, the authors trained o a 1-billion word corpus, which is 1/3 of BERT's dataset. The German training dataset consists of 500M words from mixed corpora.

4.2.2 Feature-based approach for Named Entity Recognition

As mentioned in Section 4.2.1, two separate language models are trained. To create a contextualized word representation for a word, the hidden states of both language models are extracted. From the forward language model, the hidden state after the last character of the word. The same applies to the backward language model but since it works in the reversed direction, the hidden state before the first character of the word is extracted. Concatenating both hidden states creates the contextualized word embedding. Because each language model processes text from the beginning to the respective last character, they capture both, the semantic-syntactic information of the word and the surrounding context.

The process of extracting word embeddings and passing them to the sequence labeling model is shown in Figure 4.2. It shows the sequence tagging architecture in the case of NER. Beginning with the input as sequences of characters, passed into the pre-trained bidirectional character-level language model. For each word in the sequence, the language model creates contextualized word embeddings. These word embeddings form the input into a basic, also known as vanilla, bidirectional long short-term memory (BiLSTM) with conditional random field (CRF) as final output layer. The BiLSTM produces two output states, one for the forward run and one for the reverse. The final output layer then computes the sequence likelihood over the potential labels of the tagset. For instance: The `[B-LOC]` tag indicates the beginning of a location in the chunk and the `[I-LOC]` tag if it's inside of the chunk. Therefore it is more likely that `[I-LOC]` comes right after `[B-LOC]` than `[I-PER]`. In summary, the sequence labeling model take into account the future and past input information, while the CRF output layer utilizes sentence level tag information. This combination effectively consider the surrounding context and token and therefore produces good results for the NER task [ABV18]. In addition, the authors experimented with combining Flair embeddings with different pre-trained word embeddings. Their evaluation revealed that the best performing setup is a combination of classical word embeddings and Flair embeddings. Stacking classical word-level embeddings with Flair embeddings contributes to potentially better word-level representations, as Akbik et al. [ABV18] stated.

The training process of the sequence tagging architecture on the downstream task is different to that of BERT. Whereas BERT adjusts all model parameters during training on the task, the parameters of the character-level language model are frozen. Only the weights of the sequence labeling model are adjusted during training. The authors trained their sequence labeling model on the following parameters:

- NER: ($L = 1, H = 256, B \in \{8, 16, 32\}, CG = 5, DP = var, LR \in \{0.01, 0.05, 0.1\}$)

Where L denotes the layer count, H presents the number of hidden states in the layer, B the batch size or number of examples from the overall sample. CG describes the gradient's threshold to avoid exploding gradients. Dropout probability (DP) is the likelihood that the outgoing edges of a neuron

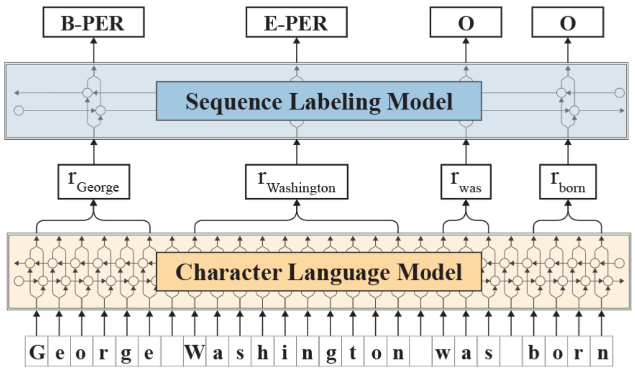

Figure 4.2: Illustration of the sequence tagging architecture of Flair. The sequence labeling model (BiLSTM-CRF) receives contextual word embeddings from the underlying character language model.

are set to zero, in this case it's chosen to be variational. Learning rate (LR) is a parameter that controls how great the weights of the network are adjusted. From this set, the authors made a model selection and chose the model with the best performance.

One year after the publication of "Contextual String Embeddings for Sequence Labeling", Akbik et al. [ABV19] proposed an enhancement and a new method for obtaining contextualized word embeddings. In their work "Pooled Contextualized Embeddings for Named Entity Recognition" they mentioned the difficulties of extracting meaningful word representations from unique words with limited context. The proposed approach addresses the issue by combining all embeddings retrieved from this word when observing the dataset. The algorithm can be defined in three simple steps:

1. Embedding the word and adding its embedding to a memory
2. Pooling operator over all embeddings in the memory
3. Concatenation of embedding from step 1 and the pooled embeddings

There are different types of pooling operators: Mean, max, min. The purpose of these operators are the selection or average, in the case of mean, of the word vectors values. With this algorithm, the pooled contextualized word embeddings ensure that both local information and global information is involved. Furthermore, the embeddings change or "evolve" over time when the same word is encountered in a different context. When training on the downstream task (e.g. NER), the memory is reset after each epoch. This method ensures the model utilizes the pooled contextualized embeddings, which are evolving over time. A promising additional feature mentioned by the authors is the continuous learning process during training and prediction phase. The approach allows the model to constantly evolve the learned embeddings when facing test data. With this publication, Akbik et al. [ABV19] presented another state-of-the-art application for sequence labeling.

4.3 Experimental Setup

This section describes the setup for the evaluation in Chapter 5. This includes the applied frameworks for the experiments and the used datasets for training and testing. To reproduce the experiments, all parameters are specified.

For the evaluation of the feature-based approach, the open-source Flair [3] framework was used. The framework enables the usage of different Flair embeddings, trained on different corpora. In addition, it is possible to load other pre-trained embeddings, like BERT embeddings or FastText embeddings. Because the evaluation will be on German datasets, mostly embeddings pre-trained on the German language were considered. Training for all feature-based approaches were performed for 150 epochs with a learning rate of 0.1 and a batch size of 32. To perform a similar experimental setup as the authors of Flair, these experiments also use the development set during training.

The experiments for the fine-tuning approach were done with the open-source framework Transformers from Hugging Face [4]. Fine-tuning different pre-trained language models was performed with the default token-classification script. Like with the Flair experiments, mostly language models pre-trained on German text were considered. For all experiments with BERT-based models the fine-tuning parameters are as follows: The experiments fine-tune for 3 epochs with a learning rate of 5e-5 and a batch size of 32. As a sequence length 128 was chosen.

[3] `https://github.com/flairNLP/flair`

[4] `https://github.com/huggingface/transformers`

5 Evaluation

The previous chapter introduced both state-of-the-art approaches for the sequence labeling task NER and mentioned the experimental setup. The upcoming chapter presents the evaluation of both approaches with the specified parameters. The evaluation is conducted on the three datasets, detailed in Table 5.1. In addition the cross-corpus experiment evaluates the performance of the best performing models, shown Section 5.3.4. Purpose of the evaluation is a comparative research of both approaches with different configurations. Furthermore, the difficulties with the various datasets and the arising complexity due to the German language are addressed.

5.1 Methods

Previous evaluations of studies or in shared tasks, like CoNLL-03 [TD03] or GermEval 2014 [BMPB15], use standard precision, recall and F_1-score metrics. Precision is the fraction of named entities that are correctly labeled by the system. The ratio of the overall entities in the corpus to the fraction recovered by the system, is referred to as recall. The formulas for precision and recall are as follows:

$$\text{Precision:} \quad \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.1}$$

$$\text{Recall:} \quad \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.2}$$

Correctly recognized entities are called True Positive (TP). FP or False Positive describes the count of tokens incorrectly labeled, whereas False Negative (FN) indicates entities which have not been recognized. To measure the models accuracy on the NER task, the F_1-score is applied.

$$\text{F}_1\text{-Score:} \quad 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{5.3}$$

It is known as the harmonic mean of precision and recall, where the maximum F_1-score value can reach one. The goal of an NER application is to maximize the F_1-score by training on the task. Since the experiments evaluate the accuracy of the model on the classification of more than two classes (Multiclass classification), the F_1-score must be determined for each class. The final score can be determined either with micro-averaging or macro-averaging. A macro-average calculates the score for each class independently and then takes the average, hence treating all classes equally. The micro-average sums up TP and FP from each individual class to compute the average score, hence the score is biased by the class frequency. It is preferable to use the micro-average in a multi-class classification, as is the case with many other academic papers and work[ABV18] [DCLT19] to take into account a class imbalance.

5.2 Datasets

The models are task-trained and evaluated on three different named entity recognition (NER) tasks. The first dataset is the German evaluation setup of the CoNLL-03 shared task [TD03]. The second dataset is from the GermEval-14 shared task [BBR14]. The original GermEval dataset includes not only spans of named entities but also a column of embedded named entities. For simplicity, the evaluation will only consider the first named entity column. The last dataset is a corpus of complaints, provided by “Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO”. The complaint dataset consists of over 170.000 tokens with a name entity proportion of 4261. Around 60% of named entities are Location entities followed by Person entities with a share of 21% and lastly Organization entities with a share of 19%. All datasets are in the “BIOES” format, as previous studies reported improvement with this tag scheme [MH16] [LBS+16].

In Table 5.1 the distribution of classifications in the three datasets, can be compared. The largest dataset is GermEval, second the German CoNLL dataset and lastly the complaint dataset. The GermEval dataset consists not only of four classifications, but also of derivations of named entities, a total of 16. For simplicity and better comparability between the datasets, derivations and classic entities of the GermEval dataset were counted together in this table. The complaint dataset was divided into 80% of the data for training and each 10% for development and testing.

Datset	LOC	PER	ORG	MISC	OTH	Total
CoNLL-03 DE	6579	5369	4441	3968	n/a	20357
GermEval-14	17275	10870	8303	n/a	4557	41005
Complaints	2757	866	638	n/a	n/a	4261

Table 5.1: List of the three datasets for the evaluation, with the respective amount of named entities in each dataset for each entity class.

The Jupyter notebooks for reproducing the experiments and the corresponding datasets, with the exception of the complaint dataset, are freely available on GitHub.[1].

5.3 Evaluation of the Results

This section covers the comparative evaluation report of the different approaches on CoNLL-03 [TD03], GermEval-14 [BBR14], and complaint dataset for the named entity recognition (NER) task. The following tables present the experimental results including the above-mentioned metrics: F_1-score, precision, and recall.

In Table 5.2 the current best published approaches are summarized. The proposed embeddings and configuration by Akbik achieves the current highest F_1-score on the CoNLL-03 NER task. With an F_1-score of 86.62, the proposed feature-based configuration *FastText + distilbert-base-german-cased* provides current state-of-the-art performance on the GermEval-14 NER task

[1] https://github.com/pascalhuszar/ner-benchmark

Approach	CoNLL-03	GermEval-14
FastText + distilbert-base-german-cased	88.09	**86.62**
best published		
Contextualized Embedding (Akbik et al. [ABV18])	**88.33**	n/a
Pooled Contextualized Embedding$_{\text{min}}$ (Akbik et al. [ABV19])	88.27	n/a
BERT-based (Ruppenhofer et al. [RRF])	84.36	n/a
BiLSTM (Riedl and Padó [RP18])	82.99	79.99
Character LSTM (Lample et al. [LBS+16])	78.76	n/a
CRF (Faruqui and Padó [FP10])	78.20	n/a
CRF (Seyler et al. [SDD+17])	77.20	n/a
BiLSTM (Akhundov et al. [ATG18])	n/a	79.21
CRF (Hanig et al. [HBT])	n/a	76.40
Multi-layer NN (Reimers et al. [RES+])	n/a	75.09

Table 5.2: Comparative evaluation of the best published approaches on the CoNLL-03 and GermEval-14 NER task. The proposed Flair embeddings by Akbik et al. [ABV18] achieve the highest F$_1$-score on the CoNLL-03 NER task. The configuration *FastText + distilbert-base-german-cased* proposed in this thesis yields the current best results on the GermEval-14 NER task.

5.3.1 Results on the CoNLL-03 Dataset

The results of both approaches for the CoNLL-03 dataset are shown Table 5.3. The overall best performance provides the feature-based approach with the configuration: *FastText + distilbert-base-german-cased*. This stacked embedding configuration consists of a word-level embedding by *fastText* and the contextual word embeddings from the model *distilbert-base-german-cased*. Stacking word embeddings describes the combination of different typs of embeddings by concatenating each embedding vector to produce the final word vector. As the authors stated, combining classical word embeddings with their proposed embeddings provides the best results [ABV18]. In fact, stacking embeddings lead to better results than a non-stacking approach, as seen in Table 5.3. The overall F$_1$-score of the feature-based approach with stacked embeddings is higher than in the fine-tuning approach. Significantly worse F$_1$-scores yield the embeddings from the uncased version of BERT models. Uncased indicates that the language model was pre-trained on a corpus with lowercase letters. For case-sensitive tasks such as NER and especially for NER for German, case information is important. The missing case information explains the significantly lower F$_1$-scores of the uncased models on CoNLL-03 dataset.

High precision values indicate a great portion of correctly labeled entities and therefore a low count of incorrectly labeled entities. Both approaches achieve relatively high precision values, except for the uncased models. For the feature-based approaches, combinations of word embeddings and contextual embeddings from BERT models have a slightly better precision than the combinations of contextual Flair embeddings and word embeddings.

A high number of recall values imply the models recognized a large numbers of named entities and therefore missed only a small number of actual named entities. As seen in Table 5.3, the recall values of the models with highest F_1-score are slightly lower than the precision values. This fact provides evidences that the models are little better in correctly labeling tokens than in recognizing tokens as named entities.

	Configuration	F_1-Score	Precision	Recall
Feature-based	Flair	82.35	84.90	79.95
	FastText(Crawl) + BytePair	83.91	86.55	81.41
	FastText + BytePair	84.06	86.74	81.54
	FastText + Flair	86.64	88.19	85.14
	FastText + Flair (History_{HA})	84.79	86.85	82.82
	FastText + Flair (History_{WZ})	84.62	86.99	82.37
	FastText + Flair (Pooled)	86.63	88.17	85.14
	FastText + bert-base-cased†	83.92	85.11	82.78
	FastText + dbmdz-bert-base-german-cased	87.85	**89.46**	86.28
	FastText + bert-base-german-cased	87.45	89.11	85.84
	FastText + bert-base-multilingual-cased†	87.10	89.09	85.18
	FastText + distilbert-base-german-cased	**88.09**	89.26	**86.95**
Fine-tuning	bert-base-cased†	76.65	79.61	73.90
	bert-base-uncased†	37.81	39.77	34.08
	bert-base-german-cased	86.03	**89.62**	82.72
	bert-base-german-uncased	50.85	52.77	49.08
	bert-base-multilingual-cased†	85.06	86.07	84.07
	bert-base-multilingual-uncased†	41.48	44.15	39.12
	dbmdz/bert-base-german-cased	**87.37**	87.45	**87.28**
	dbmdz/bert-base-german-uncased	51.84	53.78	50.05
	distilbert-base-german-cased	85.96	86.28	85.63

Table 5.3: Comparative evaluation of state-of-the-art approaches Flair and BERT with different settings. Evaluation on the German part of the CoNLL-03 dataset. Numbers in bold indicate the best result, respectively for the feature-based and fine-tuning approach.
† Word embeddings from pre-trained language models, which were trained with non-german or partly-german data.

In summary, Table 5.3 shows that the F_1-scores for both approaches are relatively high. The models, in which no distinction is made between upper and lower case, have a significantly poorer performance and are not suitable for the German sequence marking. Although the contextual word embeddings are extracted from a BERT-based system, the feature-based approach with stacked embeddings configurations yields the overall best results in the experiments. The developers of Flair stated a state-of-the-art performance of 88.33 with a different setup: *FastText + Flair*. In their experiments they repeated the experiments five times. Because of computational limitations, the experiments in Table 5.3 were only performed once.

5.3.2 Results on the GermEval Dataset

Table 5.4 presents the results on the GermEval-14 dataset, with different configurations. Although the approaches provide slightly lower scores than in the previous experiments, both systems achieve relativity high scores. The best performing model, similar to the previous experiment, is the one with the feature-based approach. The F_1-score amount to 86.62, with the setup *FastText* + *distilbert-base-german-cased*. The slightly lower score is probably related to the larger number of entity classes used in the GermEval dataset and the resulting complexities. The GermEval dataset consists of 16 different classes and therefore has four times more than the CoNLL-03 dataset.

In the experiments from Section 5.3.1, the results for the uncased version of BERT-based models indicated poor results due to the lack of attention on case-sensitivity. This is also the case for the experiments on the GermEval dataset. However, the scores are slightly better.

	Configuration	F_1-Score	Precision	Recall
Feature-based	Flair	79.83	82.12	77.67
	FastText(Crawl) + BytePair	81.52	84.29	78.93
	FastText + BytePair	81.75	84.71	78.97
	FastText + Flair	85.65	86.07	83.26
	FastText + Flair ($\text{History}_{\text{HA}}$)	83.27	85.01	81.59
	FastText + Flair ($\text{History}_{\text{WZ}}$)	82.76	84.81	80.80
	FastText + Flair (Pooled)	84.42	85.59	83.28
	FastText + bert-base-cased†	83.84	84.81	82.90
	FastText + dbmdz-bert-base-german-cased	84.55	86.17	82.99
	FastText + bert-base-german-cased	86.00	87.54	84.51
	FastText + bert-base-multilingual-cased†	83.48	85.31	81.73
	FastText + distilbert-base-german-cased	**86.62**	**87.89**	**85.39**
Fine-tuning	bert-base-cased†	78.04	77.34	78.76
	bert-base-uncased†	35.32	44.90	29.10
	bert-base-german-cased	86.18	**87.21**	85.18
	bert-base-german-uncased	52.77	56.89	49.22
	bert-base-multilingual-cased†	86.11	84.85	**87.41**
	bert-base-multilingual-uncased†	46.73	52.43	42.17
	dbmdz/bert-base-german-cased	**86.58**	85.99	87.18
	dbmdz/bert-base-german-uncased	53.45	57.48	49.95
	distilbert-base-german-cased	85.96	86.29	85.64

Table 5.4: Comparative evaluation of state-of-the-art approaches Flair and BERT with different configurations. Evaluation on the GermEval-14 dataset, using official metric (metric 1) of the GermEval 2014 Named Entity Recognition Shared Task. Numbers in bold indicate the best result, respectively for the feature-based and fine-tuning approach. † Word embeddings from pre-trained language models, which were trained with non-german or partly-german data.

5.3.3 Results on the Complaint Dataset

The results of both approaches on the complaint dataset is presented in Table 5.5. For simplicity, only the previous best performing approaches were selected. As expected, the feature-based approach provides the highest F_1-Score on this task. The best configuration is classical word embeddings stacked with the pooled embeddings of Flair. Unlike previous experiments, transformer-based models performed slightly better in the recognition of named entities. The good results for the pooled version of Flair embeddings is probably related to the functional principle. Over time, embeddings continue to evolve as they face new instances of words, and also during training on the task. This feature is presumably practical in a domain-specific environment with a small but repetitive number of named entities. Furthermore, treating words as sequences of characters, is a useful start to better handle unique and rare words out of a domain-specific corpus.

	Configuration	F_1-Score	Precision	Recall
Feature-based	Flair	68.00	68.72	67.30
	FastText + Flair	68.86	**69.64**	68.09
	FastText + Flair (Pooled)	**69.01**	69.62	**68.41**
	FastText + dbmdz-bert-base-german-cased	62.87	64.54	61.27
	FastText + bert-base-german-cased	65.57	67.97	63.33
	FastText + distilbert-base-german-cased	66.98	66.36	67.62
Fine-tuning	bert-base-cased†	55.13	57.46	52.98
	bert-base-german-cased	64.53	64.69	64.38
	bert-base-multilingual-cased†	61.95	64.14	59.91
	dbmdz/bert-base-german-cased	**65.93**	**66.78**	**65.10**
	distilbert-base-german-cased	58.50	60.88	56.29

Table 5.5: Comparative evaluation of state-of-the-art approaches Flair and BERT with different configurations. Evaluation on the complaint dataset provided by Fraunhofer IAO. Numbers in bold indicate the best result, respectively for the feature-based and fine-tuning approach. † Word embeddings from pre-trained language models, which were trained with non-german or partly-german data.

5.3.4 Results Cross-Corpus Experiment

The last experiment evaluates how well the approaches perform when trained on one corpus but tested on a different corpus. The results on this task are presented in Table 5.6. Models trained on the CoNLL dataset achieve the best F_1-score when tested on the complaints test set. Nevertheless, the results from the models trained on GermEval are only slightly smaller. Surprisingly are the precision scores in this task significantly lower than the recall scores. This means the systems can recover a good portion of overall entities out of the test set but have difficulties in correctly labeling the named entities. This probably related to the relatively small number of total entities found in the complaint corpus.

The F_1-score of the overall experiment configurations are near the "original" experiment. These findings shows that both approaches are capable to use what they have learned and apply it in other environments.

		Complaints		
Train	Configuration	F_1-Score	Precision	Recall
CoNLL	dbmdz/bert-base-german-cased	56.82	48.37	**68.85**
	FastText + distilbert-base-german-cased	**58.22**	**52.77**	64.92
GermEval	dbmdz/bert-base-german-cased	54.75	46.77	66.03
	FastText + distilbert-base-german-cased	**57.41**	**50.15**	**67.12**

Table 5.6: Comparative evaluation of the best models from the experiments in Section 5.3.1 and Section 5.3.2. Models were trained and tested on different corpora. Numbers in bold indicate the best result, respectively for the feature-based and fine-tuning approach.

5.4 Discussion

Following the experiments of the previous sections leads to a state-of-the-art model, capable of recognizing a good fraction of named entities from unstructured text. The relatively high results on the two large corpora, CoNLL-03 and GermEval-14, are probably related to the large number of total entities found in the corpus. In comparison, the best performing configuration of both approaches achieved less favorable results on the much smaller complaint dataset. Based on this result, the size of the training set could affect the performance of the models. The cross-corpus experiment revealed the different approaches and configurations are capable and robust in this scenario but still perform worse compared to relatively high results in the other experiments.

Although both approaches perform well on the named entity recognition (NER) task, the numbers are significant lower than for instance in English or Chinese. With an F_1-score of 93.50 [BEL+19] on the CoNLL-03 English set and an F_1-score of 96.72 [LSM+20] on the Chinese MSRA dataset, NER applications in other languages are capable of near-human performance [MP98]. Complex properties of the German language, for instance the capitalization of all nouns and the amount of compound words, make a proper named entity recognition difficult.

As can be seen from the results, the feature-based approaches deliver slightly better results in all experiments compared to the fine-tuning approach. The good results with the proposed character-level word embeddings by Akbik et al. [ABV18], met the expectations. Stacking word-level embeddings with contextual word embeddings from BERT was not expected to perform best in some of the experiments performed.

6 Summary and Future Work

One objective for this thesis was to investigate deep learning methods, applicable for a named entity recognition (NER) application. Chapter 3 introduced the concept of word embeddings, the idea of aligning words in multi-dimensional space to capture linguistic clues and similarities between them. Classic word embeddings can capture semantic information on word-level but struggle with rare and unique words. Another approach, character embeddings, treats words as sequences of characters and therefore better handles out-of-vocabulary (OOV) words and the notorious German compound words. Processing words not only through their structure but also as a product of their surrounding context evolved into a decisive approach and reason for state-of-the-art achievements, especially in NER for German. Language models such as BERT and Flair are able to capture profound semantic and syntactic information and therefore produce reliable contextualized word embeddings. Due to the complex properties of the German language, Flair embeddings are especially advantageous. The treatment of words on character-level seems to result in meaningful and more context-dependent representations. A suitable choice of the right sequence labeling architecture but also of a language model architecture is the key to success. Because textual data is in form of sequences it is advisable to use an architecture that exploits the sequential nature of text. A basic recurrent neural network (RNN) process words not independently but also considering the previous handled words. The challenge to maintaining long term dependencies was addressed by an LSTM unit. The "memory" cell is able to efficiently keep and process information even for longer sentences. When combining two RNN-based language models and process the input bidirectional, more contextualized word representations can be achieved. The Transformer architecture is also a promising approach for a language model and base for a German NER application. The attention mechanism enables the model to focus on relevant parts of the input while processing words and therefore produce deep contextualized word embeddings.

The Transformer architecture and the pre-training tasks of BERT made the fine-tuning approach with BERT a suitable method for German NER. The Transformer architecture in combination with the pre-training tasks of BERT forces the language model to learn contextualized representations. These contextual word embeddings then provide meaningful clues when fine-tuning on the NER task. Furthermore, the fine-tuning procedure is relatively inexpensive in terms of computing time.

Another way to address the sequence labeling task, NER, is the feature-based approach. The proposed character-level language model and sequence labeling architecture by Akbik et al. [ABV18] emerged to a suitable candidate for a German NER system. Since Flair combines the beneficial aspects of character-level embeddings and context awareness, it is capable to deal with the complexities of the German language. The stacked embeddings configuration enhance the contextual Flair embeddings with additional features from classical word embeddings or Transformer-based word embeddings.

The investigation builds the theoretical foundation for the second contribution of this thesis. The comparative evaluation of the two approaches, fine-tuning and feature-based, established an overview of the current successes in German NER. The feature-based approach consistently provided the best results on all three German datasets compared to the fine-tuning approach. Furthermore, stacking embeddings proved to be an advantageous practice. The results of the configuration fastText word embeddings and contextual word embeddings emerged to a reliable approach for NER. Nevertheless, the fine-tuning approach achieved similar but slightly worse results in the experiments. Even on a much smaller dataset both approaches achieve relatively good results, which indicates the good performance of their architecture. The cross-corpus experiment demonstrated the adaptability of both approaches. The overall F_1-Scores are only a few points below that of the "normal" experiment, suggesting that both approaches can abstract the data during training and exploit the features in a different situation.

6.1 Future Work

This thesis presented an investigation of deep learning methods for German named entity recognition (NER) and performed a comparative evaluation of two state-of-the-art approaches. The findings and conclusions motivate for further investigations and retain the incentives for latest trends in NER for German.

The pre-trained models used in the evaluation are mostly pre-trained on huge German corpora and some of them multilingual or only on English. Pre-training language models only on Germanic languages could be worth to investigate in future work. Germanic languages consist of over eight related languages, where English is the biggest and the world's most widely spoken language. Because the languages share similarities such as linguistic structure, vocabulary and grammar, they could complement each other. The multilingual configuration of BERT and Flair is an example that good results are possible if not only the target language is taken into account. The availability of a large amount of textual data and common linguistic features could be advantageous in the process of producing deep contextualized word representations for the German language.

The experiments on the complaint dataset demonstrated solid results on a relatively small dataset. The reasons for the lower results are probably the smaller number of named entities in the corpus and the domain-specific content of the dataset. To improve the performance, subsequent work may extend the dataset with more annotated data. Another future work worth investigating would be pre-training a character-level language model on the specific domain. This pre-training step could maintain and share common features between training and test data and enhance the identification of unique words.

Lastly, both frameworks are continuously updated and extended with new features. Flair recently released a new version with the possibility to fine-tune Transformer-based models. As shown in the evaluation, Transformer-based models perform well in the environment of Flair. A feature-based approach with BERT is also feasible. Devlin et al. [DCLT19] presented a feature-based experiment and achieved approximate results as with the fine-tuning approach. A thoughtful selection of BERT's layers and extracting the features may potentially yield good results for German NER.

Bibliography

[ABV18] A. Akbik, D. Blythe, R. Vollgraf. “Contextual String Embeddings for Sequence Labeling”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1638–1649. URL: https://www.aclweb.org/anthology/C18-1139 (visited on 05/04/2020) (cit. on pp. 15, 17, 19, 22, 24, 34, 35, 39, 41, 45, 47).

[ABV19] A. Akbik, T. Bergmann, R. Vollgraf. “Pooled Contextualized Embeddings for Named Entity Recognition”. en. In: *Proceedings of the 2019 Conference of the North*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 724–728. DOI: 10.18653/v1/N19-1078. URL: http://aclweb.org/anthology/N19-1078 (visited on 05/26/2020) (cit. on pp. 36, 41).

[ATG18] A. Akhundov, D. Trautmann, G. Groh. “Sequence Labeling: A Practical Approach”. In: *arXiv:1808.03926 [cs]* (Aug. 2018). URL: http://arxiv.org/abs/1808.03926 (visited on 04/24/2020) (cit. on pp. 19, 41).

[BBR14] D. Benikova, C. Biemann, M. Reznicek. “NoSta-D Named Entity Annotation for German: Guidelines and Dataset”. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 2524–2531. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/276_Paper.pdf (cit. on p. 40).

[BCB16] D. Bahdanau, K. Cho, Y. Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *arXiv:1409.0473 [cs, stat]* (May 2016). URL: http://arxiv.org/abs/1409.0473 (visited on 06/09/2020) (cit. on p. 29).

[BEL+19] A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer, M. Auli. “Cloze-driven Pretraining of Self-attention Networks”. en. In: *arXiv:1903.07785 [cs]* (Mar. 2019). URL: http://arxiv.org/abs/1903.07785 (visited on 07/06/2020) (cit. on p. 45).

[Bel03] R. E. Bellman. *Dynamic Programming*. en. Courier Corporation, Jan. 2003. ISBN: 978-0-486-42809-3 (cit. on p. 21).

[BGJM17] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov. “Enriching Word Vectors with Subword Information”. In: *arXiv:1607.04606 [cs]* (June 2017). URL: http://arxiv.org/abs/1607.04606 (visited on 05/08/2020) (cit. on pp. 17, 22).

[BKH16] J. L. Ba, J. R. Kiros, G. E. Hinton. “Layer Normalization”. In: *arXiv:1607.06450 [cs, stat]* (July 2016). URL: http://arxiv.org/abs/1607.06450 (visited on 06/11/2020) (cit. on p. 29).

[BMPB15] D. Benikova, S. Muhie, Y. Prabhakaran, S. C. Biemann. “C.: GermaNER: Free Open German Named Entity Recognition Tool”. In: *In: Proc. GSCL-2015*. 2015 (cit. on pp. 18, 39).

[BRM98] W. J. Black, F. Rinaldi, D. Mowatt. “FACILE: Description of the NE System Used for MUC-7”. In: *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*. 1998. URL: https://www.aclweb.org/anthology/M98-1014 (cit. on p. 15).

[BSF94] Y. Bengio, P. Simard, P. Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 5 (Feb. 1994), pp. 157–66. DOI: 10.1109/72.279181 (cit. on p. 27).

[CBS+15] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio. “Attention-Based Models for Speech Recognition”. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett. Curran Associates, Inc., 2015, pp. 577–585. URL: http://papers.nips.cc/paper/5847-attention-based-models-for-speech-recognition.pdf (visited on 06/09/2020) (cit. on p. 29).

[CLML18] K. Clark, M.-T. Luong, C. D. Manning, Q. Le. “Semi-Supervised Sequence Modeling with Cross-View Training”. en. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 1914–1925. DOI: 10.18653/v1/D18-1217. URL: http://aclweb.org/anthology/D18-1217 (visited on 05/08/2020) (cit. on p. 17).

[CMG+14] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *arXiv:1406.1078 [cs, stat]* (Sept. 2014). URL: http://arxiv.org/abs/1406.1078 (visited on 06/08/2020) (cit. on p. 28).

[CN02] H. L. Chieu, H. T. Ng. “Named Entity Recognition: A Maximum Entropy Approach Using Global Information”. In: *COLING 2002: The 19th International Conference on Computational Linguistics*. 2002. URL: https://www.aclweb.org/anthology/C02-1025 (cit. on p. 15).

[DCLT19] J. Devlin, M.-W. Chang, K. Lee, K. 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 (visited on 05/16/2020) (cit. on pp. 15, 17–19, 22, 24, 29, 31, 33, 34, 39, 48).

[DLS16] A. Druzhkina, A. Leontyev, M. Stepanova. “German NER with a Multilingual Rule Based Information Extraction System: Analysis and Issues”. In: *Proceedings of the Sixth Named Entity Workshop*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 28–33. DOI: 10.18653/v1/W16-2704. URL: https://www.aclweb.org/anthology/W16-2704 (visited on 11/21/2019) (cit. on p. 18).

[FIJZ03] R. Florian, A. Ittycheriah, H. Jing, T. Zhang. “Named Entity Recognition through Classifier Combination”. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 2003, pp. 168–171. URL: https://www.aclweb.org/anthology/W03-0425 (visited on 04/24/2020) (cit. on p. 18).

[FP10] M. Faruqui, S. Padó. "Training and Evaluating a German Named Entity Recognizer with Semantic Generalization". In: (Jan. 2010) (cit. on pp. 18, 41).

[HBT] C. Hanig, S. Bordag, S. Thomas. "Modular Classifier Ensemble Architecture for Named Entity Recognition on Low Resource Systems". en. In: (), p. 4 (cit. on pp. 18, 41).

[HGA+98] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, Y. Wilks. "University of Sheffield: Description of the LaSIE-II System as Used for MUC-7". In: *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*. 1998. URL: https://www.aclweb.org/anthology/M98-1007 (cit. on p. 15).

[Hoc] J. Hochreiter. "Untersuchungen zu dynamischen neuronalen Netzen". de. In: (), p. 74 (cit. on p. 27).

[HS97] S. Hochreiter, J. Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735 (cit. on p. 27).

[HSM+00] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, H. S. Seung. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit". en. In: *Nature* 405.6789 (June 2000), pp. 947–951. ISSN: 1476-4687. DOI: 10.1038/35016072. URL: https://www.nature.com/articles/35016072 (visited on 06/05/2020) (cit. on p. 27).

[JHVR15] M. Joshi, E. Hart, M. Vogel, J.-D. Ruvini. "Distributed Word Representations Improve NER for e-Commerce". en. In: *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Denver, Colorado: Association for Computational Linguistics, 2015, pp. 160–167. DOI: 10.3115/v1/W15-1522. URL: http://aclweb.org/anthology/W15-1522 (visited on 05/08/2020) (cit. on p. 17).

[KH98] G. R. Krupka, K. Hausman. "IsoQuest Inc.: Description of the NetOwl\textbackslashmbox$^\textbackslashmboxTM$ Extractor System as Used for MUC-7". In: *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*. 1998. URL: https://www.aclweb.org/anthology/M98-1015 (cit. on p. 15).

[Lan01] H. Langer. *Parsing-Experimente*. USA: P. Lang Publishing Co., 2001. ISBN: 3-631-37195-0 (cit. on p. 18).

[LBH15] Y. LeCun, Y. Bengio, G. Hinton. "Deep Learning". In: *Nature* 521 (2015), pp. 436–44. DOI: 10.1038/nature14539 (cit. on p. 15).

[LBS+16] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, C. Dyer. "Neural Architectures for Named Entity Recognition". en. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, 2016, pp. 260–270. DOI: 10.18653/v1/N16-1030. URL: http://aclweb.org/anthology/N16-1030 (visited on 05/21/2020) (cit. on pp. 23, 40, 41).

[LLA15] M. Labeau, K. Löser, A. Allauzen. "Non-lexical neural architecture for fine-grained POS Tagging". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 232–237. DOI: 10.18653/v1/D15-1025. URL: https://www.aclweb.org/anthology/D15-1025 (visited on 05/22/2020) (cit. on p. 23).

[LMP01] J. D. Lafferty, A. McCallum, F. C. N. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. ISBN: 1-55860-778-1 (cit. on pp. 15, 18).

[LPM15] M.-T. Luong, H. Pham, C. D. Manning. “Effective Approaches to Attention-based Neural Machine Translation”. In: *arXiv:1508.04025 [cs]* (Sept. 2015). URL: http://arxiv.org/abs/1508.04025 (visited on 06/09/2020) (cit. on p. 29).

[LSM+20] X. Li, X. Sun, Y. Meng, J. Liang, F. Wu, J. Li. “Dice Loss for Data-imbalanced NLP Tasks”. en. In: *arXiv:1911.02855 [cs]* (Apr. 2020). URL: http://arxiv.org/abs/1911.02855 (visited on 07/06/2020) (cit. on p. 45).

[MBXS18] B. McCann, J. Bradbury, C. Xiong, R. Socher. “Learned in Translation: Contextualized Word Vectors”. In: *arXiv:1708.00107 [cs]* (June 2018). URL: http://arxiv.org/abs/1708.00107 (visited on 05/08/2020) (cit. on p. 17).

[MCCD13] T. Mikolov, K. Chen, G. Corrado, J. Dean. “Efficient Estimation of Word Representations in Vector Space”. In: *arXiv:1301.3781 [cs]* (Sept. 2013). URL: http://arxiv.org/abs/1301.3781 (visited on 05/08/2020) (cit. on pp. 17, 22).

[MH16] X. Ma, E. Hovy. “End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF”. 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. 1064–1074. DOI: 10.18653/v1/P16-1101. URL: https://www.aclweb.org/anthology/P16-1101 (visited on 05/08/2020) (cit. on pp. 17, 40).

[MP98] E. Marsh, D. Perzanowski. “MUC-7 EVALUATION OF IE TECHNOLOGY: Overview of Results”. en. In: (May 1998), p. 71 (cit. on p. 45).

[MSC+13] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. “Distributed Representations of Words and Phrases and their Compositionality”. In: *arXiv:1310.4546 [cs, stat]* (Oct. 2013). URL: http://arxiv.org/abs/1310.4546 (visited on 05/08/2020) (cit. on pp. 17, 22).

[NDN19] K. A. Nguyen, N. Dong, C.-T. Nguyen. “Attentive Neural Network for Named Entity Recognition in Vietnamese”. en. In: *arXiv:1810.13097 [cs]* (June 2019). URL: http://arxiv.org/abs/1810.13097 (visited on 06/09/2020) (cit. on p. 29).

[NTM06] D. Nadeau, P. D. Turney, S. Matwin. “Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity”. In: *Proceedings of the 19th International Conference on Advances in Artificial Intelligence: Canadian Society for Computational Studies of Intelligence*. AI’06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 266–277. ISBN: 3-540-34628-7. DOI: 10.1007/11766247_23. URL: https://doi.org/10.1007/11766247_23 (cit. on p. 15).

[PA18] R. Panchendrarajan, A. Amaresan. “Bidirectional LSTM-CRF for Named Entity Recognition”. en. In: *Information and Computation* (2018), p. 10 (cit. on p. 17).

[PMB13] R. Pascanu, T. Mikolov, Y. Bengio. “On the difficulty of training Recurrent Neural Networks”. In: *arXiv:1211.5063 [cs]* (Feb. 2013). URL: http://arxiv.org/abs/1211.5063 (visited on 06/04/2020) (cit. on p. 27).

[PN00] J. Piskorski, G. Neumann. “An intelligent text extraction and navigation system”. In: *Content-Based Multimedia Information Access - Volume 2*. RIAO ’00. Paris, France: LE CENTRE DE HAUTES ETUDES INTERNATIONALES D’INFORMATIQUE DOCUMENTAIRE, Apr. 2000, pp. 1015–1032. (Visited on 05/04/2020) (cit. on p. 18).

[PNI+18] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer. “Deep contextualized word representations”. In: *arXiv:1802.05365 [cs]* (Mar. 2018). URL: http://arxiv.org/abs/1802.05365 (visited on 05/08/2020) (cit. on pp. 17, 23).

[PSM14] J. Pennington, R. Socher, C. Manning. “Glove: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162 (visited on 05/08/2020) (cit. on pp. 17, 22).

[QKC+09] Y. Qi, P. Kuksa, R. Collobert, K. Sadamasa, K. Kavukcuoglu, J. Weston. “Semi-Supervised Sequence Labeling with Self-Learned Features”. en. In: *2009 Ninth IEEE International Conference on Data Mining*. Miami Beach, FL, USA: IEEE, Dec. 2009, pp. 428–437. ISBN: 978-1-4244-5242-2. DOI: 10.1109/ICDM.2009.40. URL: http://ieeexplore.ieee.org/document/5360268/ (visited on 05/04/2020) (cit. on p. 18).

[RES+] N. Reimers, J. Eckle-Kohler, C. Schnober, J. Kim, I. Gurevych. “GermEval-2014: Nested Named Entity Recognition with Neural Networks”. de. In: (). URL: https://hildok.bsz-bw.de/frontdoor/index/index/year/2014/docId/285 (visited on 07/13/2020) (cit. on p. 41).

[Rös04a] M. Rössler. “Adapting an NER-system for German to the biomedical domain”. In: *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*. JNLPBA ’04. Geneva, Switzerland: Association for Computational Linguistics, Aug. 2004, pp. 92–95. (Visited on 04/24/2020) (cit. on p. 18).

[Rös04b] M. Rössler. “Corpus-based Learning of Lexical Resources for German Named Entity Recognition”. In: *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC’04)*. Lisbon, Portugal: European Language Resources Association (ELRA), May 2004. URL: http://www.lrec-conf.org/proceedings/lrec2004/pdf/373.pdf (visited on 04/24/2020) (cit. on p. 18).

[RP18] M. Riedl, S. Padó. “A Named Entity Recognition Shootout for German”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 120–125. DOI: 10.18653/v1/P18-2020. URL: https://www.aclweb.org/anthology/P18-2020 (visited on 05/04/2020) (cit. on pp. 19, 41).

[RRF] J. Ruppenhofer, I. Rehbein, C. Flinz. “Fine-grained Named Entity Annotations for German Biographic Interviews”. en. In: (), p. 10 (cit. on p. 41).

[RWC+] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. “Language Models are Unsupervised Multitask Learners”. en. In: (), p. 24 (cit. on p. 18).

[SDD+17] D. Seyler, T. Dembelova, L. Del Corro, J. Hoffart, G. Weikum. “KnowNER: Incremental Multilingual Knowledge in Named Entity Recognition”. en. In: (Sept. 2017). URL: https://arxiv.org/abs/1709.03544v1 (visited on 07/13/2020) (cit. on p. 41).

[Sie15] S. K. Sienčnik. “Adapting word2vec to Named Entity Recognition”. en. In: (2015), p. 5 (cit. on p. 17).

[STN19] S. Siami-Namini, N. Tavakoli, A. S. Namin. “The Performance of LSTM and BiLSTM in Forecasting Time Series”. In: *2019 IEEE International Conference on Big Data (Big Data)*. Dec. 2019, pp. 3285–3292. DOI: 10.1109/BigData47090.2019.9005997 (cit. on pp. 28, 31).

[TD03] E. F. Tjong Kim Sang, F. 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 - Volume 4*. CONLL ’03. Edmonton, Canada: Association for Computational Linguistics, May 2003, pp. 142–147. DOI: 10.3115/1119176.1119195. URL: https://doi.org/10.3115/1119176.1119195 (visited on 06/28/2020) (cit. on pp. 34, 39, 40).

[VC01] M. Volk, S. Clematide. “Learn-filter-apply-forget. Mixed approaches to named entity recognition”. eng. In: *Volk, Martin; Clematide, S (2001). Learn-filter-apply-forget. Mixed approaches to named entity recognition. In: 6th International Workshop on Applications of Natural Language for Informations Systems, Madrid, Spain, 2001 - 2001.* Madrid, Spain: University of Zurich, 2001. DOI: info:doi/10.5167/uzh-20271. URL: https://www.zora.uzh.ch/id/eprint/20271/ (visited on 04/24/2020) (cit. on p. 18).

[VSP+] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin. “Attention is All you Need”. en. In: (), p. 11 (cit. on pp. 18, 24, 29, 31).

[WSC+16] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, J. Dean. “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. en. In: *arXiv:1609.08144 [cs]* (Oct. 2016). URL: http://arxiv.org/abs/1609.08144 (visited on 06/18/2020) (cit. on p. 33).

[YCL+18] Y. Yang, W. Chen, Z. Li, Z. He, M. Zhang. “Distantly Supervised NER with Partial Annotation Learning and Reinforcement Learning”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 2159–2169. URL: https://www.aclweb.org/anthology/C18-1183 (cit. on p. 15).

Declaration

I declare that I have developed and written the enclosed thesis completely by myself and that I have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere. The enclosed electronic version is identical to the printed versions.

place, date, signature