FRIEDRICH-SCHILLER-
UNIVERSITÄT JENA

Fakultät für Mathematik und Informatik

Bachelor of Science (B. Sc.)'s Thesis

Data Augmentation for Named Entity Recognition in the German Legal Domain

BY

Robin Erd

born on 31.12.2000 in Schotten

Supervisors:
Prof. Dr. Birgitta König-Ries
Leila Feddoul

Heinz Nixdorf Chair for Distributed Information Systems

Jena, 22. July 2022

Kurzfassung

Die Anwendung von Named Entity Recognition auf Texte aus dem juristischen Bereich zielt darauf ab, juristische Entitäten wie Referenzen auf Rechtsnormen oder Gerichtsentscheidungen zu erkennen. Diese Aufgabe wird in der Regel mit überwachten Deep-Learning-Techniken angegangen, die große Mengen an Trainingsdaten erfordern. Vor allem für Sprachen mit geringen Ressourcen und für bestimmte Domänen sind solche Trainingsdaten jedoch oft rar. In dieser Arbeit konzentrieren wir uns auf die deutsche Rechtsdomäne, da sie für das Canarėno-Projekt von Interesse ist, das sich mit der Informationsextraktion aus und Analyse von Rechtsnormen beschäftigt. Das Ziel dieser Arbeit ist die Implementierung, Bewertung und der Vergleich verschiedener Techniken, die zur Erweiterung von verfügbaren Daten und damit zur Verbesserung der Modellleistung eingesetzt werden können. Durch Experimente mit verschiedenen Datensatzanteilen zeigen wir, dass *Mention Replacement* und *Synonym Replacement* die Leistung von sowohl rekurrenten als auch von transformatorischen NER-Modellen in ressourcenarmen Umgebungen effektiv verbessern können.

Abstract

Named Entity Recognition over texts from the legal domain aims to recognize legal entities such as references to legal norms or court decisions. This task is commonly approached with supervised deep learning techniques that require large amounts of training data. However, especially for low-resource languages and specific domains, such training data is often scarce. In this work, we focus on the German legal domain because it is of interest to the Canarėno project, which deals with information extraction from and analysis of legal norms. The objective of the work presented in this thesis is the implementation, evaluation, and comparison of different data augmentation techniques that can be used to expand the available data and thereby improve model performance. Through experiments on different dataset fractions, we show that Mention Replacement and Synonym Replacement can effectively enhance the performance of both recurrent and transformer-based NER models in low-resource environments.

Acknowledgements

I am deeply indebted to my supervisor Leila Feddoul, who supported me and gave invaluable advice and feedback while writing this thesis. Thanks should also go to professor Birgitta König-Ries and Clara Lachenmaier for the constant support and the interesting discussions that helped to continuously improve this work.

Contents

Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short-Term Memory
CRF	Conditional Random Field
DA	Data Augmentation
DL	Deep Learning
EDA	Easy Data Augmentation
ELMo	Embeddings from Language Model
FFNN	Feedforward Neural Network
FN	False Negatives
FP	False Positives
GELECTRA	(German) Efficiently Learning an Encoder that Classifies Token Replacements Accurately
HMM	Hidden Markov Model
LSTM	Long Short-Term Memory
MEMM	Maximum Entropy Markov Model
MLP	Multi-layer Perceptron
NE	Named Entity
NER	Named Entity Recognition
NLP	Natural Language Processing
POS	Part-of-Speech
RNN	Recurrent Neural Network

SVM	Support Vector Machine
TF-IDF	Term Frequency - Inverse Document Frequency
TN	True Negatives
TP	True Positives

List of Figures

List of Tables

1 Introduction

Named Entity Recognition (NER) is part of the field of Natural Language Processing (NLP). It aims to detect and classify Named Entities (NEs), such as Persons, Organizations, or Locations in text. NER is a valuable tool for information extraction and also serves as the foundation for many other NLP tasks such as machine translation [BH03], question answering [AZS06] and knowledge base construction [Etz+05]. This task is often solved by training supervised learning models which can learn complex features and solve challenging problems when provided with sufficient training data. NER training data has to be manually annotated with all class mentions that the model should later be able to detect and classify. It is hard to obtain the large amounts of annotated data required for training, as it has to be created by domain experts in a costly, tedious, and time-consuming process. This issue can be alleviated by generating new training data automatically, reducing the amount of training data initially required. This method, called data augmentation (DA), works by adding slightly modified copies of existing data to the dataset. It has its origins in the computer vision domain, where existing training images would be, e.g., rotated, cropped, or scaled to generate new training images (cf. Figure 1.1). In each domain, there exist different DA techniques. We want to evaluate NER model performance on two selected models using selected DA techniques. Our particular interest in the German legal domain directed our choice of the dataset and led us to choose the German LER dataset [LRS20]. This work was done in the context of the Canarėno (Computerunterstützte Analyse elektronisch verfügbarer Rechtsnormen) project, whose overall objective is the support of legal norm analysis done in the context of creating digital processes for administrative services.

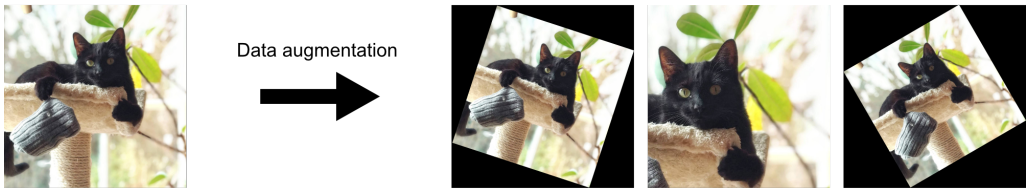


Figure 1.1: Example of data augmentation in computer vision

Thesis Scope In this work, we explore DA techniques for NER, a token-level sequence labeling task. We focus on the German legal domain, whose language is different from the language used in daily life or newspapers in many ways, e.g., structure, vocabulary, and semantics. Although data augmentation has been applied to other domains [Cha+14; Ko+15] for a long time with great success, the problem of DA for sequence labeling tasks has not been exhaustively addressed yet. There exist works applying and comparing different techniques, but to the best of our knowledge, none of them work with data from the legal domain, and all of them mainly consider English data. In addition, as far as we know, none directly compares different sources that can be used to retrieve replacements during Synonym Replacement.

The overall research goal is to evaluate and compare DA techniques for the NER task, explicitly focusing on the German legal domain. This includes surveying the data augmentation techniques applied to sequence-labeling, selecting two different state-of-the-art deep learning models, implementing three data augmentation methods, and evaluating their effect on the selected models' performance. We address the following research questions:

1. Which models are suitable for this task and the subsequent evaluation of DA techniques?
2. How well do Synonym Replacement, Mention Replacement, and Back-Translation augmentation techniques work with data from the German legal domain?
3. How do different (German) sources of replacements for Synonym Replacement compare to each other?
4. How well can Back-Translation be applied to sequence-labeling data?

Outline First, we will provide the reader with some background in Chapter 2 followed by the related work in Chapter 3. In Chapter 4 we present the concept of our work, including our goals and an overview of used techniques. We then provide details on the chosen approach and its implementation in Chapter 5. The results we achieved are then presented in Chapter 6. Chapter 7 comprises a summary of our work and contribution and an outlook on potential future research directions.

2 Background

In this chapter, we will provide information on the central concepts and techniques used in this work.

2.1 Named Entity Recognition

NER [Mar+13] was first found to be an important information extraction task during the Message Understanding Conference 6 (MUC-6) [GS96] in 1996. Since then the interest in NER has risen steadily. NER belongs to the field of NLP and is often defined as the task of detecting and classifying NEs in text (cf. Figure 2.1). To put it another way, NER aims to detect the mentions belonging to predefined semantic classes. As each word in a sentence is classified, NER is usually interpreted as a sequence labeling task. A growing number of downstream applications, such as question answering [AZS06], machine translation [BH03], and knowledge base construction [Etz+05], rely on NER. The extraction of NEs from text also allows advanced semantic search or automated highlighting of essential information. NER has been applied to the general domain (using data extracted from e.g. newspapers or Wikipedia) with generic classes like person, location, organization, and to various specific domains like the medical domain with e.g. protein, symptom, treatment, or disease as classes [Li+22; NJM21].

Figure 2.1: Example of sentence with annotated Named Entities

Traditional techniques used to solve NER were non-neural, meaning that they relied on statistical methods or hand-crafted rules which had to be carefully designed by humans [Li+22]. Due to technological advancements and progress in the field of neural networks and Deep Learning (DL), the current state-of-the-art is set by such neural methods, often combined with more traditional statistical methods.

2.1.1 Non-neural Approaches

Non-neural approaches to NER can be divided into two subcategories. In rule-based approaches, human experts come up with rules designed to recognize and classify NEs, while in learning-based approaches humans instead select the features and algorithm which then, during training, are used to generate a model that solves the task at hand. Hybrid approaches combine rule-based and learning-based techniques.

Rule-based Approaches Rule-based NER systems use external knowledge bases and syntactic-lexical patterns to recognize NEs. Although they are considered very efficient [Sha+10] and can make use of domain-specific features they are relatively expensive and non-portable. Human experts in the language, programming, and the domain have to work together to implement such a system [Saro8] and it normally cannot be used for texts in other languages, domains, or classes.

Learning Approaches Learning Approaches can be distinguished by the way they learn, which also determines what type of data they require. Supervised learning-based approaches require labeled data for training the model which can then be used to recognize and classify NEs in unseen data. For supervised learning models feature selection and engineering is a very important, time-consuming task since it determines the way in which the data is seen by the model. Such features can be orthographical, contextual, morphological, or based on the existence of the word in a list, e.g., a dictionary or lexicon. In addition to the features, the learning algorithm also has to be chosen. Popular choices [GGK18] are Hidden Markov Models (HMM), Maximum Entropy Markov Models (MEMM), Support Vector Machines (SVM) and Conditional Random Fields (CRFs) (cf. Subsubsection 2.1.2.3). In unsupervised learning, the models are not supplied with labeled data but instead have access to huge sets of unlabeled data. From these they learn how to make decisions solely based on the distributional and structural features of the data. In addition to clearly rule-based, supervised, or unsupervised learning models there also are approaches that do not fit into either of these categories. Semi-supervised learning approaches are a combination of both supervised and unsupervised models, they require a base set of labeled data and then continuously learn while working on unlabeled data.

2.1.2 Neural Approaches

Neural approaches, which are also learning approaches, can also be divided into the subcategories supervised, unsupervised and semi-supervised, as explained in Subsection 2.1.1. As noted by Li et al. [Li+22], DL techniques have three core strengths when applied to NER:

DL-based models can learn more complex features, they save significant design effort and can be trained in an end-to-end paradigm. Here end-to-end refers to the training of a potentially complicated learning system using a single model that represents the entire target system. We elaborate more on neural approaches as both models used in this work belong to this category. In this subsection, we first provide a basic understanding of Embeddings [Li+22], which are a common text representation that serves as input for machine learning models such as Long Short-Term Memory (LSTM) networks and Transformers, which are the context encoders processing the embeddings, and of the CRF layer that is often placed on top of the context encoders, acting as the tag decoder.

2.1.2.1 Embeddings

Before Embeddings were introduced, one-hot encoding was used to encode words in a manner suitable for use as input to machine learning models. In one-hot encoding each word is encoded as a binary vector with only one position being different from zero. This limits the size of the vocabulary as the dimensionality of the vectors grows proportional to the size of the vocabulary and also fails to capture semantic relations between words. Embeddings are a way to represent categorical data as vectors in a continuous space. This allows for convenient operations such as the measuring of the distance between two categorical variables. We identified three main types of embeddings: Word Embeddings, Character Embeddings, and Contextual Embeddings.

Word Embeddings Traditional word embeddings like GloVe [PSM14], word2vec [Mik+13] and fastText [Gra+18] represent each word as a vector in a continuous vector space so that the distance between words relates to their similarity. They are typically learned using unsupervised machine learning methods on huge corpora with billions of words. Similar representations are learned for words that frequently appear close to each other. In a neural network architecture such embeddings are used via an embedding layer that functions like a lookup table, mapping words to their corresponding vector representation.

Character Embeddings Character embeddings, first employed by Santos et al. [SG15], capture information about the lexical composition of words. They are generally considered less rich in semantics than word embeddings and therefore mostly used to enhance word embeddings. The character embedding representation of two words where just one letter differs is almost identical, making them useful for handling misspellings, slang words, or other out-of-vocabulary words. For training character embeddings there exist two commonly used architectures: Convolutional Neural Networks (CNNs) [SZ14] and Recurrent Neural

Networks (RNNs) [Lam+16]. An advantage of character embeddings is that their vocabulary size is limited by the size of the alphabet, making them relatively lightweight.

Contextual Embeddings In contrast to the embeddings covered in the previous paragraphs, the representation of words in contextual embeddings depends on the surrounding context of the word. Contextual Embeddings successfully capture the difference between polysemous words, e.g. "left" (as a noun in "to the left", as a verb in "he left" and as an adjective in "the left house"), meaning that the same word can have different embeddings depending on its context. In the following paragraphs, two popular such approaches are described.

Contextual Word Embeddings Peters et al. [Pet+17], was one of the first to introduce context-sensitive word embeddings with TagLM, also called Pre-ELMo. Later the contextual language models ELMo [Pet+18] and BERT [Dev+19] were established. BERT and other recent models make use of Transformers that read the entire input sequence of words at once, allowing them to return a vector representation of each token that also considers its context.

Contextual String Embeddings Contextual string embeddings were introduced by Akbik et al. [ABV18]. They built a bidirectional Long Short-Term Memory-based (BiLSTM) character-level language modeling architecture [HS97] that interprets sentences as sequences of characters and is trained with the objective to predict the next character in the current sequence.

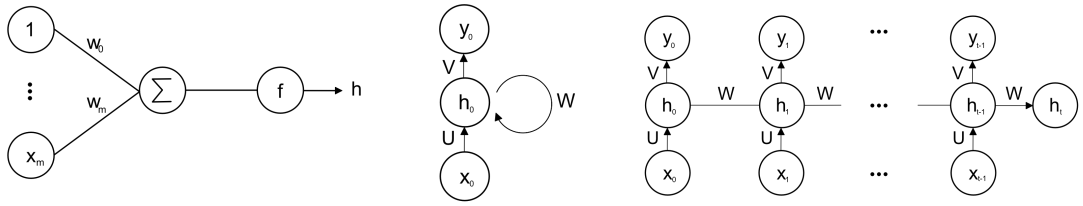
2.1.2.2 Neural Networks

In the following, we will explore different neural network architectures and their usage in NER.

Feedforward Neural Networks Feedforward Neural Networks (FFNN), also known as Multi-layer Perceptron (MLP), are the most basic form of neural networks and were invented by Frank Rosenblatt in the 1950s and 1960s [Ros61]. They consist of artificial neurons that are organized into layers[1], where the outputs of one layer are the inputs of the next layer. These connections, so-called edges, are associated with weights, which make up the parameters that are learned during training. Each of the artificial neurons works as a function, taking in a number of inputs x_i, multiplying the inputs by their corresponding weight w_i, summing the products, and applying an activation function $f(x)$ (cf. Equation 2.1). A very simple FFNN consisting of one neuron is depicted in Figure 2.2a.

$$h(x_1, \ldots, x_m) = f(w_0 + w_1 x_1 + \ldots w_m x_m) \tag{2.1}$$

[1]The layers between the input and output layers are referred to as hidden layers. The size of a layer is defined by the number of nodes it contains.



(a) Simple FFNN **(b)** (Folded) RNN **(c)** Unfolded RNN

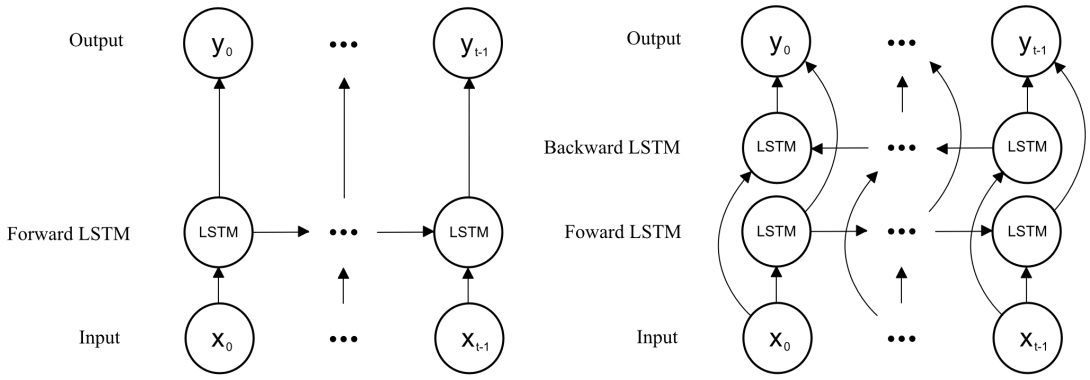
Figure 2.2: Simple FFNN, (folded) RNN, and unfolded RNN architectures

Recurrent Neural Networks RNNs [Elm90] advance traditional FFNN by capturing time dynamics through their recurrent connections (cf. Figure 2.2b). Their output y_i is not only a function of the momentary input features x_i but also of their last state h_{i-1}. This enables them to memorize information and make predictions conditioned on past information. In Figure 2.2b and Figure 2.2c U contains the weights of the edges connecting the input with the hidden state, V contains the weights of the edges connecting the hidden state with the output and W contains the weights of the edges connecting the current hidden state with the previous hidden state. RNNs can also be visualized by unfolding[2] them with respect to their input sequence (cf. Figure 2.2c). In theory, the architecture of RNNs allows them to learn complex long-distance relationships and patterns, but in practice, the most recent inputs dominate the current state of the RNN.

Long Short-Term Memory Networks LSTM networks [HS97] were designed to overcome the limitations of pure RNNs. Instead of directly using the last state as input, they use an LSTM unit (cf. Figure 2.3a) in which several gates control which information gets memorized and which is forgotten. These gates are learned during training and allow LSTM networks to successfully capture long-distance dependencies and patterns [GSC99]. The state of the LSTM after having sequentially processed some k elements is often referred to as the context.

Often one does not only have access to past elements but also future elements - this is also the case during NER. It is inherent to the structure of not only unidirectional LSTM (cf. Figure 2.3a) but also RNNs in general that they only have access to the context comprised of the past inputs. To make use of both enclosing contexts, the past and the future, Huang et al. [HXY15] were the first to apply an architecture called BiLSTM to NLP tagging datasets (cf. Figure 2.3b). The input sequence is provided to two different LSTM networks simultaneously, forward and backward, respectively. Consequently, the hidden state of the forward LSTM contains the past context, and the hidden state of the backward LSTM contains the future

[2] The terms "folded" and "unfolded" only refer two different interpretations.



(a) Unidirectional LSTM architecture **(b)** Bidirectional LSTM architecture

Figure 2.3: Unidirectional and bidirectional LSTM architectures

context. The combined context can then be obtained by simply concatenating both hidden state vectors, $y_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$.

Transformers Transformers [Vas+17] are a novel architecture for solving sequence-to-sequence tasks that also handle long-range dependencies very well. In sequence-to-sequence tasks, an input sequence is turned into another sequence. They use a self-attention mechanism to produce representations of their input- and output sequences and are built based on an encoder-decoder architecture. The encoder consists of one Multi-Head-Attention Layer followed by an FFNN and the decoder also has those two layers and an additional Masked Multi-Head Attention layer on top.

RoBERTa, introduced by Liu et al. [Liu+19], is an optimized variant of BERT [Dev+19], which works based on the Transformer architecture. The training procedure was adjusted by removing the Next Sentence Prediction task and introducing dynamic masking. It was pre-trained in an unsupervised manner on 160 GB of unlabeled English text using the Masked Language Modeling objective, where part of a sentence is masked, and the model tries to predict the masked words. Models trained on such tasks can be fine-tuned to solve a variety of other NLP tasks that use whole sentences as input, such as question answering, sequence classification, or sequence labeling, achieving state-of-the-art results. Typically such models take raw text as input, not requiring preprocessing such as tokenization. XLM-RoBERTa [Con+20] is a multilingual version of RoBERTa, pre-trained on 2.5 TB of multi-lingual data.

Model Training Neural machine learning models are trained using variations of Back-propagation algorithms. *Backpropagation* [RHW86] is a training algorithm that updates the weights of the edges connecting the neurons in the network with the goal to minimize the error compared to the expected output for the corresponding input. The algorithm consists

of the following steps. First, the model is presented with a training input pattern which is propagated through the network to get an output. Then, to calculate the error, the predicted output is compared with the expected output. Finally, the derivative of the error with respect to the network weights is calculated and the weights are adjusted to minimize the error, also called loss (calculated using loss functions [Wan+22]). This process is then repeated. One pass of the above-mentioned process through the complete dataset is called an *epoch*. The *learning rate* determines how rapidly the model learns, i.e., the strength with which the weights are adjusted after each epoch. It can be changed dynamically during training.

Different strategies for adjusting the weights of a neural network during training are called *optimizers*. One very popular type of optimizer is the Stochastic Gradient Descent [Net19], which is a stochastic approximation of the gradient descent optimization algorithm. Instead of updating the model weights after calculating the loss and gradient for the entire dataset, the weights are updated after calculating the loss and gradient for only a few samples at a time. This number of samples considered when calculating the loss is referred to as the *mini-batch size*. It controls how frequently the weights are updated during training. There exist many variations of the SGD with additional parameters, such as, i.e., *momentum*, which can be useful to reduce the variance and soften the convergence.

Backpropagation Through Time [Wer90] is a variant of Backpropagation adapted to the structure of RNNs. To obtain the dependencies of the model variables and parameters from the recursive structure, the computational graph of the RNN is unfolded regarding the RNNs input length (cf. Figure 2.2c). Then the loss is calculated and accumulated for each timestep. A big drawback of Backpropagation Through Time is that the computation of the derivatives contains many multiplications involving the weight matrix, causing the gradient to either explode or nearly vanish.

One popular approach to solving this issue is applying *gradient clipping* [Zha+20]. This introduces a threshold that gradients cannot exceed, essentially clipping them by scaling them down so that their norm matches the set threshold. There are two more common practices applied to improve model training; learning rate annealing and adding a dropout layer.

Learning rate annealing [Nak+21] refers to the idea of starting training with a relatively high learning rate and then progressively decreasing it. There are different techniques for determining how much and when to decrease the learning rate, the most popular of which is step decay. With step decay, the learning rate is reduced by a set percentage at fixed epoch intervals. Other popular approaches are to reduce the learning rate depending on e.g. the error or micro F1-score on the development or the training split.

Dropout [Sri+14] on the other hand is rather a part of the model itself. A dropout layer ignores the output of a set of randomly selected neurons during training, not forwarding it to the next layer. These neurons are also not considered during the following backward pass.

There are multiple ways of deciding when to stop model training. It can be stopped after completing a predetermined fixed number of epochs, when the error falls below a predetermined value, when the learning rate reaches zero, or when the error on the development split, which may be computed during training after each epoch, starts increasing again.

2.1.2.3 Conditional Random Field Layer

CRFs [LMP01] are statistical models used in sequence labeling tasks. In contrast to other classifiers that predict the class of each input independently in linear CRFs, each prediction depends on its direct neighbors. CRF layers are often added to work as decoders on top of an LSTM model, making the output sequence of the underlying model the input sequence of the CRF layer (cf. Figure 2.4). The learned parameters of a CRF layer are stored in a state transition matrix that contains the probabilities of transitioning from one state to another. In NER this corresponds to e.g. the probability of a B-PER tag (designating the first token of a person-type NE) being followed by a E-PER tag (designating the last token of a person-type NE). Tagging schemes are explained in more detail in Subsection 2.2.2.

Using CRFs as the decoding layer of a NER system makes it possible to take into account the relations between adjacent labels during prediction. This improves the overall model performance as, following the tagging schemes, some label sequences are invalid (e.g. a token tagged with B-PER being followed by a token tagged with I-LOC).

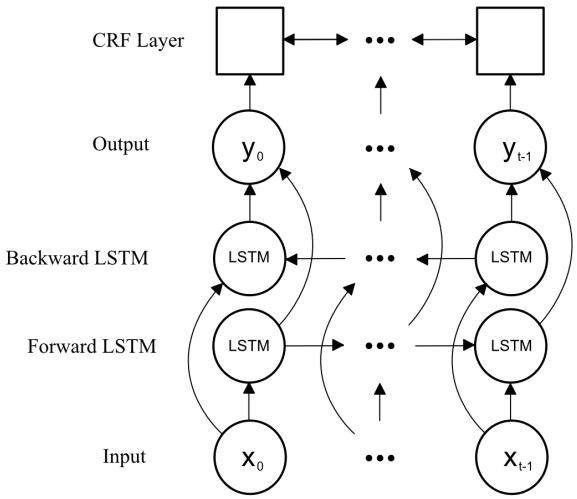


Figure 2.4: BiLSTM-CRF architecture

Table 2.1: List of annotated datasets for English NER, (non-exhaustive)

Dataset	#Tags	Source
MUC-6 [Sun96]	7	Wall Street Journal
CoNLL'03 [SM03]	4	Reuters news
ACE 2005 [Wal+06]	7	Transcripts, news
OntoNotes 5.0 [Wei+13]	18	Magazines, news
HYENA [Yos+12]	505	Wikipedia
GENIA [Pro]	36	Biological, clinical text
W-NUT 2017 [Der+17]	6	User-generated

Table 2.2: Example of sentence tokenization with different tokenizers

Tokenizer	Instance
None	`'Alex is going to Berlin.'`
Whitespace Tokenizer	`['Alex', 'is', 'going', 'to', 'Berlin.']`
SoMaJo Tokenizer [PU16]	`['Alex', 'is', 'going', 'to', 'Berlin', '.']`

2.2 Data

As mentioned in Subsection 2.1.1 the type of data required by a machine learning model depends on the model's architecture and task. Training supervised NER models requires tagged corpora of sentences with annotated class mentions, many of which were created (cf. Table 2.1). These datasets are usually tagged using a common format (cf. Subsection 2.2.2).

2.2.1 Tokenization

A crucial step in processing data during and in preparation of NLP sequence-labeling tasks is tokenization [WK92]. This describes the process of splitting a character sequence into pieces, transforming text into a structured, more predictable form that is suitable for further use. It is common to first split a document into sentences which are then split into lists of tokens. Besides words, special characters, punctuation, or other atomic units are often considered single tokens. Many tokenizers work based on predefined sets of rules, dictionaries, or regular expressions, all of which are specific to the language the tokenizer is to be used with. There are several open-source tools available to perform tokenization. When talking about tokens in this work, we refer to tokens generated by a word-level tokenizer (cf. Table 2.2), but there also exist subword-level tokenizers.

2.2.2 Tagging Schemes

Tagging schemes, also referred to as segment representation techniques, define what and how annotated information is stored. During tagging in NER, each token of a sentence gets assigned its corresponding tag that is part of the chosen scheme combined with the class indicator (cf. Table 2.3). Besides NER, Part-of-Speech (POS) tagging [Sch94] is a popular sequence labeling task. Different existing schemes (cf. Table 2.4) were compared by Konkol and Konopík [KK15]. We will use the terms 'tag', 'label' and 'annotate' synonymously.

- **IOB1** In the IOB1 scheme, also referred to as BIO, tokens that are the beginning of a known NE get tagged with B, tokens inside of entities with I, and all other tokens with O, as they are outside of any known entity.
- **IOB2** This scheme is identical to the IOB1 scheme with the exception that the first token of entities that directly follow an entity of the same class is also tagged with B, which is not the case in IOB1.
- **IOE1** This scheme also is very similar to the IOB1 scheme, with the change that it indicates the end of entities with an E instead of indicating their beginning.
- **IOE2** This scheme is identical to the IOE1 scheme with the same modifications as IOB2. The last token of entities that are directly followed by an entity of the same class is also tagged with E, which is not the case in IOE1.
- **IOBE** The IOBE scheme is a combination of the IOB2 and IOE2 scheme. The beginning of each known NE is tagged with B and its last token with E, while tokens inside of it are tagged with I.
- **IOBES** This scheme is a further extension of the IOBE scheme that makes the information more explicit, labeling single-token entities with S.

Table 2.3: Example of IOB2-tagged NER sentence

Alex	is	going	to	Los	Angeles	in	California	.
B-PER	O	O	O	B-LOC	I-LOC	O	B-LOC	O

2.3 Data Augmentation

The data used to train machine learning models has to be of high quality and as diverse as possible while also representing the data that the system might encounter during its application as accurately as possible. Supplying models with more such training data improves

Table 2.4: List of tagging schemes, (non-exhaustive)

Name	Scheme
IOB1	Inside, Outside, Begin
IOB2	Inside, Outside, Begin
IOE1	Inside, Outside, End
IOE2	Inside, Outside, End
IOBE	Inside, Outside, Begin, End
IOBES	Inside, Outside, Begin, End, Single

overall performance [Bow+15] and generalization capabilities of the model, makes them more robust and is an important measure to prevent overfitting [JK15].

It is hard to obtain the annotated data required for supervised training because it has to be created by domain experts in a costly, tedious, and time-consuming process. Therefore there is a constant need for more annotated data, especially domain-specific data. One approach to solving this problem, besides semi-supervised learning and transfer learning, is DA. Among these, DA is the most versatile as any supervised learning algorithm could be applied to the augmented data. The goal of DA is to apply transformations to the already existing data while retaining the semantic meaning, consequently increasing the total amount of data.

While there has been ample research concerning DA in other machine learning domains such as computer vision [Cha+14] or speech recognition [Ko+15], there are relatively few works available dealing with DA in the field of NLP and consequently NER. The following non-exhaustive list provides an overview of DA techniques applicable to data used for sequence-labeling tasks that we found during the literature review. Table 2.5 shows the effect of each DA technique applied to an example sentence. For token-wise techniques such as Random Deletion or Synonym Replacement, a parameter $p \in [0, 1]$ determines how many tokens relative to the total length of the sequence are transformed.

Synonym Replacement This technique describes the replacement of one or more tokens in the original token sequence with their synonyms, including tagged tokens. It does not result in a change in the original syntactic pattern and is a rather limited but easily applicable technique. The source of the synonym may be a pre-trained static word embedding (e.g. GloVe [WY15]), a dictionary (e.g. WordNet [WZ19; ZZL15]), a contextual language model [Wu+19] or other sources that store suitable information. In many cases, multiple synonyms are collected, and then the most similar, according to cosine word similarity, is chosen.

Mention Replacement This technique describes the replacement of one or more entities (tagged mentions) in the original token sequence with an entity of the same class appearing in the training data [RM17] and does not result in a change in the original syntactic pattern.

Instead of using a dictionary containing the entities that occur in the training data, other suitable sources may also be used. An extension of this technique is to not only replace entities, but any token with a token of the same label appearing in the training data, including tokens outside of entities (Label-wise token replacement) [DA20]. We will be using the terms 'mention' and 'entity' interchangeably from now on.

Random Deletion This technique describes the random deletion of one or more tokens from the original token sequence, excluding the first tokens of entities [WZ19]. There exist variants where tokens annotated as entities are fully exempted from deletion.

Random Insertion This technique describes the random insertion of one or more tokens into the original token sequence [WZ19]. There exist different variants regarding the position of the inserted token(s). The inserted token can be taken from one of the sources discussed in *Synonym Replacements*. With the use of language models, not only single tokens but entire sequences can be inserted at the beginning or end of a sentence.

Random Swap This technique describes the random swap of two or more tokens of the original token sequence [WZ19]. There exist different variants regarding groups of tokens that can be swapped. One variant is to partition the original sequence into segments composed of tokens with the same label and then shuffle the tokens within these segments [DA20].

Generative Approaches Generative approaches gained in popularity with the advance of contextual language models such as BERT [Dev+19]. Among the generative approaches, there exist three main directions. The generation of training examples using (1) knowledge bases and sentence patterns or schemes [KBC20; Kun+20], (2) using contextual language models [Din+20; Zho+21] or (3) through interpolation or extrapolation of existing training examples [ZYZ20].

Noising Techniques Blank noising [WZ19]., unigram noising [WZ19], keyboard error injection, and spelling error injection may be classified as noising techniques. They can be used to make a model more robust in dealing with exactly these irregularities which often occur in data extracted from social media or similar domains.

TF-IDF based Word Replacement Term Frequency - Inverse Document Frequency [Xie+20] is an unsupervised DA method that replaces less-relevant tokens (e.g. "this") with other less-relevant tokens (e.g. "a"), that are identified based on the calculation of term frequency and the inverse document frequency.

Back-Translation Back-translation [SHB16] is a technique that uses translators and their effect on the syntactic structure and wording of sentences. To transform a sentence, the original sentence is first translated into a second language and then translated back into

the original language. Variants of this method exist where not only one but two or more intermediate languages are used.

Table 2.5: Example of data augmentation techniques applied to sentence with (potential) changes highlighted in blue color.

Technique	Instance						
None	Alex S-PER	met O	John S-PER	in O	Berlin S-LOC	. O	
Synonym Replacement	Alex S-PER	saw O	John S-PER	in O	Berlin S-LOC	. O	
Mention Replacement	Elon B-PER	Musk E-PER	met O	John S-PER	in O	Berlin S-LOC	. O
Random Deletion	Alex S-PER	met O	John S-PER		Berlin S-LOC	. O	
Random Insertion	Alex S-PER	met O	saw O	John S-PER	in O	Berlin S-LOC	. O
Random Swap	met O	Alex S-PER	John S-PER	in O	Berlin S-LOC	. O	
Generative Approaches *here: using a pattern:	Emma S-PER <person>	met O <verb>	Olivia O <person>	in O in	Austin S-LOC <location>	. O .	
Noising Techniques	Alex S-PER	met O	John O	inn O	Berlin S-LOC	. O	
TF-IDF *depends on complete dataset	Alex S-PER	met O	John S-PER	at O	Berlin S-LOC	. O	
Back-Translation	Alex S-PER	met O	with O	John S-PER	in O	Berlin S-LOC	. O

2.4 Evaluation Metrics

For the evaluation of the performance of NER systems, different measures have been proposed and used [NS09]. All methods have in common that they are based on the comparison of the generated annotations with annotations which were performed by human experts (the gold standard). Generally, we are not only interested in the performance of the system across all entity classes but also in single classes. The task of NER, as described in Section 2.1 comprises two subtasks, (1) detecting the entity and its boundaries and (2) classifying the entity into one of the given label classes. Both tasks are commonly evaluated together.

Exact-match evaluation only considers a NE to be recognized correctly if it is an exact match of the ground-truth annotation, regarding both type and span. *Partial-match evaluation* considers

a NE to be recognized correctly if the entity was assigned the correct type and the span of the entity in the ground truth and the span as identified by the model overlap. There are more complex evaluation schemes, used by e.g. ACE [Dod+04], where each NE class is weighted, and during evaluation partial matches and nested NEs are taken into account.

Accuracy describes the percentage of correct predictions relative to all predictions. It is calculated using Equation 2.2, where true positives (TP) are the (number of) correctly identified NEs, true negatives (TN) are tokens outside of entities that were recognized as such, false positives (FP) are the incorrectly labeled NEs, and false negatives (FN) are the missed NEs. The accuracy is not a suitable metric to capture classification performance due to the class imbalance of NER datasets - most tokens usually do not belong to an entity. Still, it can be useful to report the accuracy metrics class-wise as part of a confusion matrix. Precision and Recall, as well as the F1-score, are also calculated based on TP, TN, FP, and FN counts.

$$\text{accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.2}$$

Precision (cf. Equation 2.3) is defined as the ratio of true positives relative to the sum of the true positives and false positives. Put simply, the percentage of actually positive out of the positive-predicted. *Recall* (cf. Equation 2.3), also referred to as sensitivity, can be described as the ratio of true positives relative to the sum of true positives and false negatives. It reports the percentage of true positives that are actually found by the model. The *F1-score* (cf. Equation 2.4) is the harmonic mean of precision and recall.

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = 2 \times \frac{TP}{TP + TN} \tag{2.3}$$

$$\text{F1-score} = 2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{2.4}$$

Precision, recall, and F1-score can be calculated in two ways, either micro- or macro-averaged. The *macro average* is computed by computing the precision, recall, and F1-score for each class and then averaging them by dividing by the number of classes. Macro-averaging admits equal importance to all classes. The *micro average* on the other hand is computed by aggregating the TP, TN, and FP counts of all classes and then calculating recall, precision, and F1-score. This admits each sample the same importance so that frequently occurring classes in the dataset have a higher impact on the final metrics than rare classes. Grandini et al. [GBV20] provide a comprehensive overview and explanation of common multi-class-classification metrics.

3 Related Work

In this chapter, we survey previous works concerned with DA for NER training data and provide an overview of other studies applying NER to texts from the German legal domain.

3.1 Named Entity Recognition

As most of the state-of-the-art research in NLP and specifically NER is done in the English language there exist few works regarding NER in the German legal domain. Glaser et al. [GWM18] applied three different techniques for the extraction of entities from German legal contracts, namely GermaNER[1], DBpedia Spotlight [Men+11], and templated NER. They achieved F1-scores of 0.80, 0.87, and 0.92 respectively, on a dataset composed of 25,423 tokens for GermaNER and DBpedia Spotlight and 7,790 tokens for templated NER. They conclude that templated NER poses a suitable solution as long as templates exist. Leitner et al. [LRM19] compared the performance of different BiLSTM-CRF model architectures in NER on a newly created dataset of German legal documents comprising 66,723 sentences taken from 750 German court decisions that were published online. Their BiLSTM-CRF models with character embeddings outperform the other configurations, with the BiLSTM-CRF with character embeddings from an RNN achieving an F1-score of 0.9595 on coarse-grained classes and 0.9546 on fine-grained classes. More recently, Zöllner et al. [Zöl+21] compared the effect of using different pre-training techniques for small BERT models on their performance and, in addition to that, also presented modifications of the fine-tuning process that result in performance improvements. During the evaluation, they used, among others, the (coarse-grained) German LER dataset [LRS20] and achieved an F1-score of 0.9488 by fine-tuning with a CRF extended by an additional NER-rule.

Brugman et al. [Bru18] performed NER on a selection of manually annotated publicly available Dutch court rulings, aiming to anonymize names and extract information such as case and punishment. They employed BiLSTM-CRF and Transformer architectures and achieved

[1] GermaNER is a CRF-based Named Entity Tagger.

F1-scores ranging from 0.8184 to 0.8794. Badji et al. [Bad18] explored Legal Entity Extraction with NER Systems on newly created English and Spanish corpora, which were manually annotated beforehand. They achieved an F1-score of 0.95 employing a rule-based system and an F1-score of 0.75 employing another system comprised of a combination of different services.

Aside from the legal domain, the German CoNLL 2003 NER dataset [SM03] comprised of annotated texts from German news is frequently used as a benchmarking dataset and was also used by the following works. In 2018 Akbik et al. [ABV18] proposed Contextual String Embeddings and achieved an F1-score of 0.8832, improving on the previous best of 0.7876 set by Peters et al. [Pet+18]. Later Schweter et al. [SA20] achieved an F1-score of 0.8834 using a fine-tuned XLM-RoBERTa Transformer model.

3.2 Data Augmentation

Replacement The replacement of words with words similar in meaning was one of the first techniques to be employed for DA and, due to its simplicity, effectiveness, and robustness, still frequently is.

Zhang et al. [ZZL15] applied WordNet-based Synonym Replacement to eight different text classification datasets. Müller et al. [MT16] applied WordNet-based Synonym Replacement, increasing the size of the SICK dataset from 5, 000 training examples to 15, 000. This improved the relatedness score of their model achieved in the sentence similarity scoring task by 0.04 compared to their baseline of 0.8422 that they established by training only on the original SICK dataset. Instead of using explicit knowledge bases like WordNet, words with similar meanings can also be extracted from word embeddings such as GloVe, word2vec, or fastText. Wang et al. [WY15] applied word2vec-based Synonym Replacement to a newly created Twitter dataset used for topic classification. They expanded the training data to five times the original size and achieved an F1-score increase of 0.024 compared to their baseline of 0.341.

As contextual language models were introduced and achieved new performance records in many domains of NLP due to their powerful representations of context they also started to be used for DA. Wu et al. [Wu+19] randomly replaced one to two words with a [MASK] token, which was then filled by a label-conditioned contextual language model[2]. This doubled the number of available training examples and led to an F1-score increase of 0.0195 for their employed RNN model, compared to their baseline of 0.7743.

[2] The prediction has to be label-conditioned because otherwise, the text class may change.

There also exist DA approaches for NER tasks that focus on generating new training instances by modifying only the tagged NEs in sentences (Mention Replacement). Raiman and Miller [RM17] augmented the SQuAD [Raj+16] question answering training set by swapping NEs in the training data with others of the same type retrieved from an external knowledge base. Thereby they increased the number of training examples by 10% and improved the F1-score by 0.001, compared to their baseline of 0.781. Liu et al. [Liu+20] randomly replaced entities in the training data with entities from a dictionary that did not appear in the training data to increase the size of their newly created NER dataset comprised of papers from ACL and ACM. They also applied an Entity Masking technique where they replaced entities with randomly generated words. The best performance was achieved by masking entities in 20% of the original data, leading to an increase of the F1-score by 0.0755 compared to their baseline of 0.6716.

Combined Techniques Wei et al. [WZ19] presented four techniques now known as easy-data-augmentation (EDA) techniques, namely Synonym Replacement, Random Insertion, Random Swap, and Random Deletion. They evaluated the augmented dataset on five benchmark text classification tasks and demonstrated that EDA techniques, used in combination as well as individually, can improve the model performance. Generating four additional sentences per original sentence by randomly changing 10% of the original sentence, they improved the average F1-score across the datasets by 0.009 compared to their baseline of 0.874. Kang et al. [Kan+20] extended the EDA techniques proposed by Wei et al. by adding an external knowledge-based system and modifying them to work with NER tasks by preserving a valid tagging sequence[3]. They achieved improvements in a medical NER task (F1-score +0.02, baseline 0.68) on the EBM-NLP corpus [Nye+18] and also tested a domain-specific augmentation method with which they achieved greater improvements. Issifu et al. [IG21] adapted and simplified the modified EDA techniques used by Kang et al. They evaluated the techniques on two biomedical NER datasets and achieved an improvement of 0.0086 in F1-score on the NCBI dataset [DLL14] compared to their baseline of 0.8669 and no change in performance on the Species-800 dataset [Paf+13] with a baseline of 0.7298. Sabty et al. [Sab+21] evaluated the effect of applying Synonym and Mention Replacement, modified EDA techniques, and Back-Translation to Arabic-English Code-Switching NER data[4]. They achieved an increase of 0.0151 in F1-score compared to their baseline of 0.7769 after applying Back-Translation in combination with Synonym and Mention Replacement, which raised the number of available training examples from 5,306 to 10,612. Dai et al. [DA20] applied Label-wise Token Replacement, Synonym Replacement, Mention Replacement, and Shuffle Within Segments

[3] They do so by, e.g., preventing the deletion of words tagged as the beginning of an entity.
[4] Code-Switching refers to text containing more than one language in the same sentence.

techniques[5] to the MaSciP (materials science) [Mys+19] and i2b2-2010 (biomedical) [Uzu+11] NER datasets. By augmenting the MaSciP dataset and training an RNN-based model, they achieved a decrease of 0.004 by applying Mention Replacement and an increase of 0.003 by applying Synonym Replacement with a baseline of 0.764. When using a Transformer-based model with the same datasets, they achieved an increase of 0.002 by applying Mention Replacement and a decrease of 0.001 when applying Synonym Replacement with a baseline of 0.798. They find that, on average, for Transformers, Synonym Replacement works best, while for RNNs, Mention Replacement is most effective. Shim et al. [Shi+20] applied EDA techniques with the modification of performing Random Noise Insertion instead of Random Insertion to increase the size of a new text classification dataset created based on a web survey about sleep issues. They achieved an increase in macro F1-score of 0.01 (baseline 0.68) when using the full augmented dataset comprised of 14, 364 original sentences and 71, 377 synthetic sentences, a decrease of 0.01 (baseline 0.65) when using and augmenting 50% of the original dataset, resulting in 7, 182 original sentences and 34, 059 synthetic sentences and an increase of 0.21 (baseline 0.39) when using and augmenting 10% of the original dataset, resulting in 1, 436 original sentences and 7, 043 synthetic sentences. Yaseen et al. [YL21] applied Label-wise Token Replacement, Synonym Replacement, Mention Replacement, and Shuffle Within Segments techniques to the MaSciP and Species-800 datasets. By augmenting the MaSciP dataset and training a BiLSTM-CRF model using GloVe embeddings, they achieved an increase in F1-score of 0.0096 when applying Mention Replacement and an increase of 0.00158 when applying Synonym Replacement compared to a baseline of 0.7537.

Back-Translation Back-translation was first introduced by Sennrich et al. [SHB16] and used to augment Neural Machine Translation training data. It is also often used for other sentence-level problems such as topic classification [Xie+20], sentiment analysis [Luq19], and question answering [Yu+18]. Yu et al. [Yu+18] applied Back-Translation and Mention Replacement to the SQuAD question answering dataset, resulting in training data roughly three times the original size. Using this dataset led to an increase of 0.011 in F1-score with a baseline of 0.827. Yaseen et al. [YL21] applied segment-wise Back-Translation to the MaSciP and Species-800 datasets and achieved an increase in F1-score of 0.0645 (Species-800, baseline 0.6044) and 0.0148 (MaSciP, baseline 0.7537) using GloVe embeddings with a BiLSTM-CRF model. Sabty et al [Sab+21] applied Back-Translation to Arabic-English Code-Switching NER data, resulting in a decrease in performance. With a baseline of 0.7769, the F1-score dropped by 0.2828 when using a Neural Machine Translation model and by 0.006 when using Google Translate. However, as mentioned earlier, combined with Synonym and Mention Replacement they achieved an increase of 0.0151 in F1-score.

[5] Shuffle Within Segments is a variant of Random Swap in which token swaps only take place within segments of tokens with the same label.

Although many studies have investigated the effect of DA techniques, it is difficult to draw universal conclusions. The works apply different techniques, use different dataset sizes from varying domains and employ different models during evaluation. Still, many works found that DA is more effective for small datasets than for large [DA20; IG21; Shi+20; YL21].

4 Concept

This chapter aims to provide an overview of the used dataset and the NER models used together with their configurations. We also present the selected DA techniques as well as how we plan to implement and evaluate them.

4.1 Data

We evaluate the DA methods on the German LER dataset [LRS20] from the legal domain. It contains approx. 67,000 sentences with over 2 million tokens which are classified into 19 fine-grained semantic classes and 7 coarse-grained classes. The sentences were sourced from court decisions from 2017 and 2018 published online by the Federal Ministry of Justice and Consumer Protection. The distribution of entities is listed in Table 4.1 based on Leitner et al. [LRS20]. We decided to use this dataset as we are particularly interested in the German legal domain. For our work, we decided to use the 17 fine-grained classes.

Leitner et al. do not provide train-dev-test splits, so we shuffle the data and split it ourselves (cf. Table 4.2). Note that we only apply DA on the training data and leave the development and test splits untouched. This is due to some forms of DA potentially introducing unwanted correlations between train and test split if generated examples are placed in different splits. Suppose we split the dataset after performing the augmentations, e.g., Mention Replacement. In that case, this could lead to a sentence being part of the train and test split with only the entity being different, making the prediction task much easier. We work with IOB2, the tagging scheme in which the data is provided. When working with the data during augmentation, the tokenization of the original sentence should be reproduced. We, therefore, use the SoMaJo tokenizer [PU16] used by Leitner et al. across all our code.

4.2 NER Models

To evaluate the effect of the different approaches of DA, we chose two model architectures that reflect the current state-of-the-art and established commonly used architectures.

BiLSTM-CRF (cf. Subsubsection 2.1.2.2 and Subsubsection 2.1.2.3) We decide to use the default model of the FLAIR framework presented by Akbik et al. [Akb+19]. FLAIR is a framework designed specifically to facilitate the training and general use of sequence labeling and text classification models. The default model consists of a BiLSTM network with a CRF tag decoder. We follow Akibk et al. [ABV18] by training this architecture using *Stochastic Gradient Descent without momentum*, *clipping gradients* at 5, and *annealing the learning rate* by halving it if score on the development set does not increase for 5 consecutive *epochs*. We perform model selection over the *learning rate* $\in \{0.01, 0.05, 0.1\}$ and the *mini-batch size* $\in \{8, 16, 32\}$, choosing the model with the best micro F1-score on the development set. For other parameters, we follow Reimers et al.'s [RG17] analysis of hyperparameters of BiLSTM models in sequence labeling and use variational[1] *dropout*, 256 hidden states (neurons) per layer, and set the number of hidden layers to 1. As Akbik et al. [ABV18] recommend combining traditional word embeddings with Contextual String Embeddings we use German fastText embeddings [Gra+18], German forward Flair embeddings [ABV18] and German backward Flair embeddings [ABV18]. We stop training if the learning rate falls below 0.0001 or we reach a maximum number of epochs, which we set to 150, again following Akbik et al. [ABV18].

Fine-tuned XLM-RoBERTa (cf. Subsubsection 2.1.2.2) In 2020 Schweter et al. [SA20] presented FLERT, an extension of the aforementioned FLAIR framework, which adds support for Transformer-based approaches. We will make use of the fine-tuning approach as Schweter et al. showed that it outperforms the feature-based approach, in which the Transformer is only used to generate embeddings which are then again used as input to a BiLSTM-CRF model. We choose the XLM-RoBERTa Transformer model over models trained specifically for the German language as preliminary studies showed that it achieves better results on the LER dataset than, e.g., GELECTRA [CSM20]. Following Akbik et al. [SA20], we train this architecture using the AdamW *optimizer* [LH19], with a fixed small number of epochs as stopping criterion [Con+20], and a very small learning rate. The learning rate is increased from 0 to 5e−6 during the warmup phase and then linearly decreases until it reaches 0 by the end of the training. We perform model selection over the maximum number of epochs $\in \{10, 20\}$ and the mini-batch size $\in \{1, 4, 8\}$, again choosing the model achieving the best micro F1-score on the development set. We use the commonly used subword pooling strategy "first" [Dev+19]

[1] In variational dropout, in contrast to regular dropout, the same dropout mask is used for inputs, outputs and recursive connections in the network over the course of one timestep [GG16].

in which the representation of the first subtoken is used as a representation for the entire token. This is necessary as the contextual language model internally uses subword-tokens, but we require word-token predictions of the NER task.

4.3 Data Augmentation

We test the effect of Synonym Replacement and compare different German sources of replacements, which, to the best of our knowledge, has not been done yet. We also assess the impact of Mention Replacement and explore a new variant of applying Back-Translation to sequence-labeling data. In contrast to Yaseen et al. [YL21], we aim to apply Back-Translation to complete sentences, including entities, not only segments without entities. We chose Synonym Replacement and Mention Replacement techniques due to their previous successful application to data used for NER tasks and because they are likely to preserve the syntactic and semantic correctness of the sentences. We explore Back-Translation as we believe it has great potential to reliably and robustly introduce sentences with syntactical changes. We decided against DA techniques that involve randomly adding or removing words as we assume they are more likely to alter the meaning and degrade the correctness of a sentence than replacement techniques. We also do not consider complex generative approaches (cf. Section 2.3) as e.g. training a conditional contextual language model [Din+20] is a very expensive process and approaches based on interpolation [ZYZ20] go beyond the scope of this work.

Synonym Replacement During Synonym Replacement, we replace a percentage of non-tagged tokens in sentences with a replacement that is similar in meaning (cf. Table 4.3). To qualify for replacement, a token must not be part of a tagged mention and must match a regular expression pattern. This assures that we do not try to replace, e.g., punctuation marks or numbers. We compare three different sources of replacements: OpenThesaurus [Nab05], fastText embeddings, and a contextual language model (XLM-RoBERTa). In addition, we also test different replacement percentages (20%, 40%, 60% of qualified non-tagged tokens per sentence).

Mention Replacement In Mention Replacement, we replace entities in sentences with entities from the same class extracted from the corpus (cf. Table 4.4). We always replace all mentions in a sentence, and the replacement is drawn randomly from an entity-class dictionary generated based on the training data.

Back-Translation Back-Translation refers to the translation of a sentence into another language and the subsequent back-translation into the source language. The goal is to cause

syntactical (and structural) changes in the sentence while leaving its semantics unaffected (cf. Table 4.5). This is possible because different languages vary in, i.e., word order, and back-translating often preserves this change caused by the initial translation. When using Back-Translation to augment a sentence, we first extract the entities and their type. Then the full sentence is translated to the intermediate language, also referred to as the pivotal language, and back to German. As pivotal language, we decide to use English as we assume that this ensures high-quality translations. We then try to map the extracted entities back to the sentence based on their token sequence. For this to work, sentences must not contain a token sequence multiple times with different label sequences each. Otherwise we will not be able to distinguish between the two identical token sequences and assign the correct label to each (cf. Table 4.6 and Table 4.7). The augmentation is considered a success if all extracted entities are found in the back-translated sentence; otherwise, the back-translated sentence is discarded (and only the original sentence is kept).

Evaluation We test each of the three DA techniques independently. The proposed evaluation workflow is illustrated in Figure 4.1 and consists of the following steps. We establish a baseline by taking the train split of the German LER dataset ①, taking a fraction from the complete training split ②, training both models (BiLSTM-CRF, Fine-tuned XLM-RoBERTa) with this non-augmented data ④, and evaluating the performance of each model on the test split ⑤. As fractions we decided for (%) : {1, 10, 30, 50, 100} of the complete training split. To evaluate the impact that DA techniques have on the model performance, we then apply the selected DA techniques to these fractions ③, train on the generated augmented fractions ④ and evaluate the impact that DA has on the performance of the models in the NER task relative to the original dataset size ⑤. We use the micro F1-score for model performance evaluation (cf. Section 2.4). We decided against using the macro F1-score due to the class imbalance in the German LER dataset (cf. Table 4.1). Unfortunately, we cannot report the average of multiple runs and the related standard deviation due to time constraints.

Besides assessing DA's impact on NER model performance, we also evaluate the DA itself. We do this by manually examining the quality of the generated examples, randomly sampling a set of sentences, and rating their syntactical and semantic correctness as well as how different they are compared to the original.[2]

[2]As this is a time-costly procedure we only do so for selected configurations (replacement percentage, replacement source) of the Synonym Replacement technique.

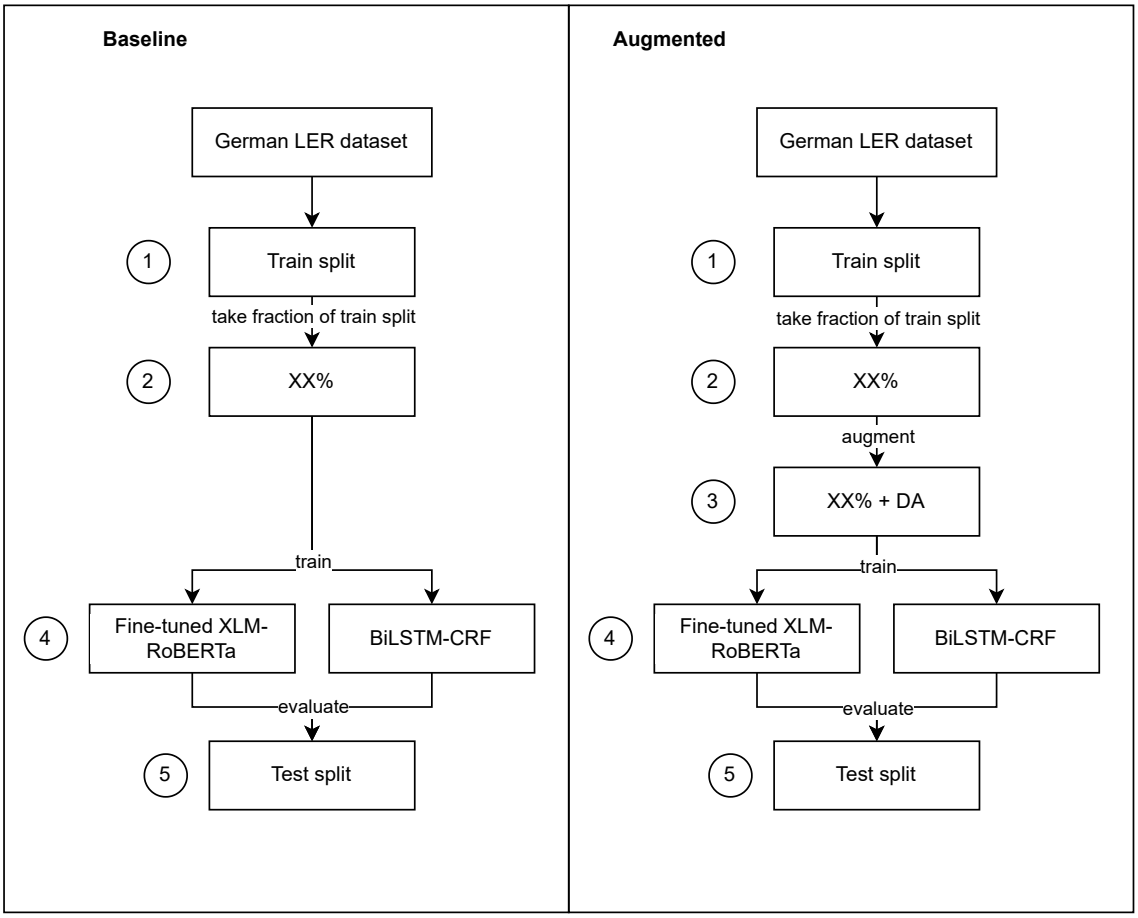


Figure 4.1: Overview of the proposed approach

Table 4.1: Distribution of fine-grained (f) and coarse-grained (c) classes in the German LER dataset based on Leitner et al. [LRS20]

Class Type	Id	Name	Description	#Tokens	%Tokens
f	1	PER	Person	1,747	3.26
f	2	RR	Judge	1,519	2.83
f	3	AN	Lawyer	111	0.21
c	**1**	**PER**	**Person**	**3,377**	**6.30**
f	4	LD	Country	1,1429	2.66
f	5	ST	City	705	1.31
f	6	STR	Street	136	0.25
f	7	LDS	Landscape	198	0.37
c	**2**	**LOC**	**Location**	**2,468**	**4.60**
f	8	ORG	Organization	1,166	2.17
f	9	UN	Company	1,058	1.97
f	10	INN	Institution	2,196	4.09
f	11	GRT	Court	3,212	5.99
f	12	MRK	Brand	283	0.53
c	**3**	**ORG**	**Organization**	**7,915**	**14.76**
f	13	GS	Law	18,520	34.53
f	14	VO	Ordinance	797	1.49
f	15	EUN	EU legal norm	1,499	2.79
c	**4**	**NRM**	**Legal norm**	**20,816**	**38.81**
f	16	VS	Regulation	607	1.13
f	17	VT	Contract	2,863	5.34
c	**5**	**REG**	**Case-by-c. regul.**	**3,470**	**6.47**
f	18				
c	**6**	**RS**	**Court decision**	**12,580**	**23.46**
f	19				
c	**7**	**LIT**	**Legal literature**	**3,006**	**5.60**
Total				**53,632**	**100**

Table 4.2: List of dataset split sizes

Split	#Sentences	%
train	46,706	70
dev	10,008	15
test	10,009	15

Table 4.3: Example of sentence augmented with Synonym Replacement

Alex	is	going	to	Los	Angeles	in	California	.
Alex	was	walking	towards	Los	Angeles	around	California	.

Table 4.4: Example of sentence augmented with Mention Replacement

Alex	is	going	to	Los	Angeles	in	California	.
Chloe	is	going	to	Mexico	in	United	Kingdom	.

Table 4.5: Example of sentence augmented with Back-Translation with English as pivotal language

Original:	Alex	geht	nach	Los	Angeles	in	Californien	.
Intermediate:	Alex	goes	to	Los	Angeles	,	California	.
Result:	Alex	geht	nach	Los	Angeles	,	Californien	.

Table 4.6: Example of sentence satisfying the multiple-sequence-label condition

2016	fuhren	A.	,	F.	und	Z.	gemeinsam	nach	F.	.
O	O	B-PER	O	B-PER	O	B-PER	O	O	B-PER	O

Table 4.7: Example of sentence not satisfying the multiple-sequence-label condition

Vom	...	führte	das	...	ein	Einsatznachbereitungsseminar	in	...	durch	.
O	O	O	O	O	O	O	O	B-ST	O	O

5 Implementation

This chapter provides details on the implementation of the NER models and DA techniques.

5.1 Machine Learning Models

As mentioned in Chapter 4 we choose two models, namely BiLSTM-CRF and fine-tuned XLM-RoBERTa to evaluate the impact of different DA techniques. This sections build on the concepts and methods related to model training explained in Subsubsection 2.1.2.2.

5.1.1 BiLSTM-CRF

We use the FLAIR framework [Akb+19] to implement a BiLSTM network with a CRF decoder layer. This architecture corresponds to the `SequenceTagger` class configuration listed in Table 5.1. It is trained using the `ModelTrainer` (cf. Table 5.2). To implement the combination of German fastText embeddings and German Flair embeddings we use the `StackedEmbeddings` class which combines the embeddings by concatenating each embedding vector to form the final word vector. For training, we set the `embeddings storage mode` to "gpu" whenever possible. It controls where the embeddings get stored during training. We are able to use this with all except the 100% + DA datasets, as they do not fit into the available GPU memory[1]. This avoids the need to repeatedly shuffle data from the CPU to the GPU and improves efficiency. We set the `patience` parameter to 5. This controls after how many consecutive epochs without increasing development score the learning rate is halved.

Following the FLAIR documentation we set the `ModelTrainer` parameter `write_weights` to `True` and use the default configuration of the remaining `SequenceTagger` and `ModelTrainer` parameters. We decided against using the configuration that performed best in model selection and instead chose a marginally worse configuration as the time cost was disproportionate (an improvement in micro F1-score of only 0.0063 points (from 0.9626 to 0.9632) for 17.6h of

[1] When not storing the embeddings in the GPU we request 96GB of (CPU) memory.

training instead of 7.5h). We make this decision as we have to train the model more than 75 times to evaluate the impact of the different DA techniques.

5.1.2 Fine-tuned XLM-RoBERTa

We use the FLERT extension [SA20] of the FLAIR framework [Akb+19] to implement a Transformer-based sequence tagger. To fine-tune a transformer on the NER task, a single linear layer is added to the Transformer, in our case the XLM-RoBERTa model (XLM-R). Then the entire architecture is trained on the desired task. This architecture can be built through the configuration of the `SequenceTagger` (cf. Table 5.1) which is fine-tuned using the `ModelTrainer` (cf. Table 5.2). Note that we set the `Hidden size` parameter that controls the number of neurons per layer to `256` because FLAIR requires it to be non-zero, although `Use RNN` is set to `False`, so no hidden layer is added. As mentioned in Section 4.2 we configure the `TransformerWordEmbeddings` class to use the subword pooling strategy `first`. Its configuration is listed in Table 5.3. We set the `layer` parameter to `-1`, indicating that we want the output to be from the topmost layer of the Transformer. We set `use context` to `False`, as we do not want to make use of document context, as we want the model to treat each sentence independently. As the GPU we use can handle the selected mini-batch size, we do not make use of mini-batch chunking, which would further split mini-batches into chunks, causing slow-downs. Following the FLAIR documentation we set the `SequenceTagger` parameter `reproject_embeddings`[2] to `False` and use the default configuration of the remaining `SequenceTagger` and `ModelTrainer` parameters. Table 5.2 and Table 5.1 list the parameters we use in all experiments with the fine-tuned XLM-R model. Again we decided against using the configuration that performed best in model selection and instead chose a marginally worse configuration as the time-cost was disproportionate (an improvement in micro F1-score of only 0.0063 points (from 0.9687 to 0.9691) for 28.8h of training instead of 14.5h).

Table 5.1: SequenceTagger configuration

Parameter	BiLSTM-CRF	XLM-R
Hidden size	256	256
Dropout	0.25	-
Embeddings	de-fasttext, Flair	Transformer
Tag Type	ner	ner
Use RNN	True	False
Use CRF	True	False
Reproject embeddings	Default (True)	False

[2] This adds an additional linear layer on top of the embedding layer.

Table 5.2: ModelTrainer configuration

Parameter	BiLSTM-CRF	XLM-R
Embeddings storage mode	gpu	-
Learning rate	0.05	5.0e-6
Mini-batch size	32	1
Patience	5	-
Max epochs	150	10

Table 5.3: TransformerWordEmbeddings configuration

Parameter	Value
Model	xlm-roberta-large
Layers	-1
Subtoken pooling	first
Fine-tune	True
Use context	False

5.2 Data Augmentation Techniques

The following section provides a detailed explanation of the implementation of each augmentation technique. After the application of each technique to a corpus, the generated sentences, as well as the combined original and generated sentences, are saved. During each augmentation, a log file containing information on the changes in each sentence and statistics regarding the entire process is generated. Generally, the techniques work on a sentence level. The tokenizer we use is the SoMaJo tokenizer configured as shown in Table 5.4. As we were not able to find information about the used configuration by Leitner et al. [LRS20] we tested all possible configurations and use the configuration in which the tokenization of most sentences was reproduced correctly.

5.2.1 Synonym Replacement

Our Synonym Replacement function has two main parameters, the percentage of (qualified) tokens to be changed in a sentence and the replacement source. During the augmentation of a dataset, the following steps are taken as illustrated in Figure 5.1. First, we check whether we can reproduce the way the sentence is tokenized from its plain sentence representation. If this is not the case the sentence has to be skipped and cannot be augmented. If a sentence passes this check we collect the indices of all not tagged tokens in the sentence. These are the tokens eligible for replacement which are checked against the regular expression pattern

Table 5.4: SoMaJo tokenizer configuration

Parameter	Value	Description
Language	de-CMC	sets the language
Split sentences	False	perform sentence-splitting in addition to tokenization
Split camel case	False	split words written in camelCase

`r"[a-zA-ZäöüÄÖÜß]+"`, matching a sequence of one or more characters from inside the brackets[3]. Consequently, this filter rejects all tokens containing, among other things, punctuation, numbers, whitespace, or special characters. We then shuffle this list to assure a random replacement and multiply its length with the percentage of tokens that should be replaced (Step 2). We take the floor of the result to get the integer number of desired replaced tokens. We chose to take the floor because this leads to the replacement percentage being an upper bound on the percentage of tokens changed per sentence, which we believe is more convenient[4] than a lower bound. It also refers to the percentage of eligible tokens to be changed, excluding e.g. punctuation. We then iterate through the shuffled list of not tagged tokens and substitute them with their replacement until the calculated number of desired replaced tokens is reached. For replacing a token in a sentence we pass the token that is going to be replaced to a method that returns a suitable replacement from the selected source. All sources return a list of replacement candidates which are filtered with the same regular expression as mentioned earlier to prevent e.g. punctuation from being inserted into the middle of the sentence (this occasionally happens with fastText embeddings as replacement source) (Step 3). Should the candidate list contain the original token it is removed, ensuring that the replacement retrieved from the chosen source differs from the original token.

To get candidates from the thesaurus we set up a MySQL database created from a dump (May 30, 2022) of the OpenThesaurus database. Originally we used the publicly available API but due to the limit of 60 requests per minute we decided to set up our own instance on a `db-f1-micro` unit (with 10GB SSD storage) on Google Cloud SQL[5]. To retrieve replacement candidates from the MySQL database we use the `mysql-connector-python` package[6] and execute the query provided by the OpenThesaurus documentation [7]. However, our code still provides the option to use the public API, whose XML response is processed using the `beautifulsoup4` package[8]. The results returned by the OpenThesaurus are not ranked based on a scoring function. This can be considered a limitation of this source. Retrieving candidates using the fastText

[3] We use the regular expression full-match method.
[4] This presumes that too much change can have a negative impact.
[5] https://cloud.google.com/sql/
[6] https://pypi.org/project/mysql-connector-python
[7] https://www.openthesaurus.de/about/download
[8] https://pypi.org/project/beautifulsoup4

embeddings is simple and also comes with a meaningful ranking, as the python `fastText` package[9] includes a built-in method to get the four nearest neighbors, which are then filtered and returned as possible candidates, ordered by their distance to the original token. In contrast to the other two sources the contextual language model, XLM-RoBERTa, requires the entire sentence as a plain string to return replacement options. In this string, we replace the token that should be replaced with the `<mask>` token and let the contextual language model solve the fill-task, returning a list of candidates ranked by the score that the contextual language model assigns to each candidate. This is done step-by-step for each replacement candidate and implemented using the `fill-mask` pipeline provided by the `transformers` package[10] [Wol+20].

In the default configuration, the first candidate from the filtered list is chosen as replacement. If the chosen source does not return a replacement candidate that passes the filter (regular expression) we abort the replacement of the current token and continue with the next token from the list. Having found a suitable replacement, we tokenize it and insert it in place of the original token. Due to the filtering of the replacements, we only consider the ones that consist of a single token. Consequently, we do not have to shift any labels in the sentence, as we replace a single token with another single token (Step 4). The labels of the tokens to the left and right of the replaced token can simply be copied and the replacement token is annotated with the ”O” tag, as the replacement for a token that is not an entity will also not be an entity.

5.2.2 Mention Replacement

The Mention Replacement function does not have parameters but expects an entity dictionary as input, besides the dataset to augment. The entity dictionary is built from the entities contained in the training dataset. It contains lists of all entities occurring in the training dataset grouped by their classes. During the augmentation of a dataset, the following steps are taken as illustrated in Figure 5.2. As before, we verify if the tokenization can be reproduced. We then get the indices of all entities as well as their corresponding class and store them in a list (Step 2). Next, we iterate through the list of entities, randomly drawing an entity of the same class from the dictionary as replacement (Step 3). We tokenize the replacement and calculate the offset that it causes in the original sequence of tokens and their corresponding labels. We then replace the tokens that are part of the original entity with the tokens that comprise the replacement entity. We also add the offset to the indices of all remaining entities in our entity list that are located to the right of the replaced entity so that the indices now refer to the entity’s position in the modified sentence (Step 4). The annotations of the tokens to the left and right of the replaced entity can simply be copied. Finally, we annotate the

[9] https://pypi.org/project/fasttext
[10] https://pypi.org/project/transformers

replacement with the same class the original entity had, deducing the correct IOB2-scheme tags from its length. We do not enforce that the replacement retrieved from the dictionary differs from the original entity[11].

5.2.3 Back-Translation

Our Back-Translation function does not have any parameters, as we hard-coded the pivotal language to be English. When starting the augmentation of a sentence, we first have to verify the reproducibility of its tokenization, as we did for Synonym Replacement and Mention Replacement. During the augmentation of a dataset, the following steps are taken as illustrated in Figure 5.3. First, we collect all the entities, their starting index, and their class into a list (Step 1). Then, before continuing, we check if the token sequences that comprise entities occur multiple times in the sentence (Step 2). As explained in Section 4.3 we can only augment a sentence if these sequences have the same label sequence at each occurrence. We then back-translate and tokenize each entity we extracted from the sentence to increase the chance of being able to fully re-annotate the back-translated sentence (Step 3). We then back-translate the full sentence as a plain string, using English as pivotal language, and tokenize the returned new sentence (Step 4). Finally, we try to find each extracted entity in the back-translated, tokenized version of the sentence by matching the token sequence of either the original or the back-translated version of the entity with a sequence of tokens in the back-translated sentence, so that we can label it (Step 5). We annotate all matching token sequences with the corresponding label[12], as, due to the previously mentioned constraint, we can be sure that in case there are multiple matching token sequences all matches have the same label sequence. If we are unable to find all extracted entities in the back-translated sentence the augmentation of the sentence is aborted. After labeling all entities (extracted from the original sentence) in the back-translated sentence, the process should be considered finished since the translation is not introducing new entities. We can therefore annotate all remaining tokens with "O".

By default our implementation of Back-Translation uses the `BackTranslation` package[13] which makes use of the `googletrans`[14] package and the Baidu Translation API[15] to provide free translation functionality. However, our code also provides the option to use the Google Cloud Translation API[16] or the DeepL API[17] if the required API keys are provided.

[11] Unfortunately this only came to our attention after performing augmentation and evaluation.

[12] We annotate either all occurrences of the original token sequence or the back-translated token-sequence, depending on which occurs more frequently in the back-translated sentence.

[13] https://pypi.org/project/BackTranslation

[14] https://pypi.org/project/googletrans/3.0.0

[15] https://api.fanyi.baidu.com

[16] https://cloud.google.com/translate

[17] https://www.deepl.com/pro-api

Figure 5.1: Synonym Replacement process by example

STEP 1

Original sentence

Token	§	11	Abs.	2	EigZulG	erlaubt	Änderungen	der	von	§	11	Abs.	1	Satz	1	angeordneten	Einheit	.
Label	B-GS	I-GS	I-GS	I-GS	I-GS	O	O	O	O	B-GS	I-GS	I-GS	I-GS	I-GS	I-GS	O	O	O
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

STEP 2

Extract entity information

Original Entity	§ 11 Abs. 2 EigZulG	§ 11 Abs. 1 Satz 1
Class	GS	GS
Start Index	0	9

STEP 3

Get first replacement

§ 11 Abs. 2 EigZulG → Dictionary → § 34 Abs. 2 Nr. 1 —tokenize→

§	34	Abs.	2	Nr.	1

STEP 4

Insert and label first replacement

Token	§	34	Abs.	2	Nr.	1	erlaubt	Änderungen	der	von	§	11	Abs.	1	Satz	1	angeordneten	Einheit	.
Label	B-GS	I-GS	I-GS	I-GS	I-GS	I-GS	O	O	O	O	B-GS	I-GS	I-GS	I-GS	I-GS	I-GS	O	O	O
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Calculate offset

len(replacement) - len(original_entity) = 6 - 5 = 1

Update indices of remaining entities

Original Entity	< done >	§ 11 Abs. 1 Satz 1
Class		GS
Start Index		9**+1 = 10**

STEP 3

Get second replacement

§ 11 Abs. 1 Satz 1 → Dictionary → § 7 Abs. 4 UmwRG —tokenize→

§	7	Abs.	4	UmwRG

STEP 4

Insert and label second replacement

Token	§	34	Abs.	2	Nr.	1	erlaubt	Änderungen	der	von	§	7	Abs.	4	UmwRG	angeordneten	Einheit	.
Label	B-GS	I-GS	I-GS	I-GS	I-GS	I-GS	O	O	O	O	B-GS	I-GS	I-GS	I-GS	I-GS	O	O	O
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Figure 5.2: Mention Replacement process by example

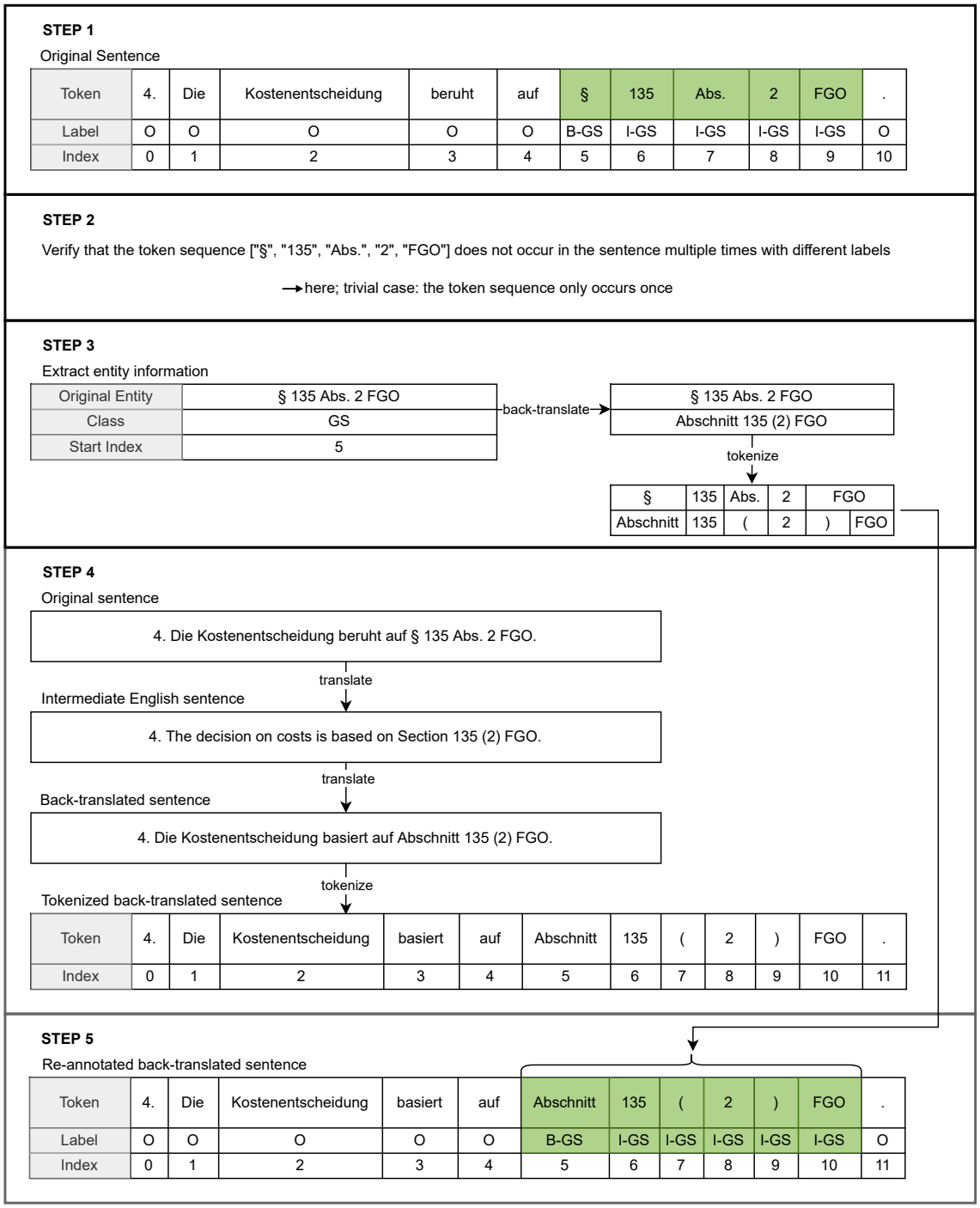


Figure 5.3: Back-Translation process by example

6 Evaluation

In this chapter, we present and discuss the results of the experiments we performed.

6.1 Baselines

Table 6.1a lists the baseline performances to which we will compare the performance of both models trained with augmented datasets. The results show that in very low-data settings, represented by the 1%-dataset, the BiLSTM-CRF model outperforms the XLM-R model. For all other dataset fractions, the XLM-R model outperforms the BiLSTM-CRF model. We also notice that we already achieve relatively good performances with only 10% and 30% of the original dataset, possibly due to its large size. The training duration of both models is listed in Table 6.1b. Training and evaluation are run on a single NVIDIA A100 GPU (cf. Table 6.2).

Table 6.1: Baseline training duration and evaluation results

(a) Baseline results, micro F1-scores

Dataset	BiLSTM-CRF	XLM-R
1%	0.6959	0.6089
10%	0.9071	0.9130
30%	0.9356	0.9416
50%	0.9489	0.9559
100%	0.9572	0.9661

(b) Training duration in minutes

Dataset	BiLSTM-CRF	XLM-R
1%	82	49
10%	108	128
30%	207	302
50%	246	481
100%	505	918

6.2 Data Augmentation

As mentioned in Section 4.1 and Section 5.2 we can only augment sentences whose tokenization is reproducible. This constraint affects 4, 233 of the original number of sentences, corresponding to 9.06% of all training data, which is consequently not eligible for augmentation. The following subsections provide details on the results achieved after training our models on

Table 6.2: List of detailed implementation specifications

Item	Specification
CPU	Intel(R) Xeon(R) Gold 6242R CPU @ 3.10GHz
GPU	NVIDIA A100, 40GB memory
Graphic driver	Nvidia graphic driver version 465.19.01
CUDA	Version 11.3
OS	AlmaLinux 8.3
Python	Python 3.9.12
Pytorch	Version 1.11.0

the augmented datasets and evaluating the quality of the resulting augmented data for each augmentation technique. Note that we state percentage improvements relative to the baseline performance. In consequence, we consider a change in F1-score from e.g. 0.10 to 0.11 a 10% improvement.

6.2.1 Synonym Replacement

The time Synonym Replacement takes depends on the replacement percentage and used source. It varies from only 0.41 seconds per sentence using OpenThesaurus in combination with a replacement percentage of 20% to 12 seconds per sentence using XLM-RoBERTa as source in combination with a replacement percentage of 60%. In consequence, the augmentation of the German LER dataset using Synonym Replacement took between 5 and 155 hours, resulting in between 35, 673 and 40, 875 new generated sentences (cf. Table 6.3). Generally, the number of successfully augmented sentences increases with the replacement percentage. This technique can boost the dataset size by up to 87.3%. The maximum number of augmented sentences was generated with a replacement percentage of 60% and the contextual language model and fastText embeddings as the source. As the replacement percentage is an upper bound, we also track how many tokens relative to all tokens were replaced. Here, fastText and XLM-RoBERTa perform very well, returning a replacement for almost all tokens qualified for replacement. OpenThesaurus can only return a replacement for on average 60% of the proposed tokens. We also notice that around 50% of all tokens are eligible for replacement, i.e., pass the regular expression filter (cf. Subsection 5.2.1). In turn our replacement-percentages of 20%, 40% and 60% results in around 10.8%, 22.7%, and 34.6% of tokens being attempted to be replaced. Table 6.4 lists our results after training both models on the datasets augmented with different configurations. In the following, we will discuss the results of each replacement source. It also contains the average change in micro F1-score across all datasets for each replacement source and percentage combination.

Table 6.3: Dataset sizes after applying Synonym Replacement with either OpenThesaurus (THE), fastText embeddings (FTX), or a contextual language model (CLM) as replacement source and different replacement percentages

Dataset	Source	Original	20%	40%	60%
1%	THE	468	815	853	862
10%	THE	4671	8,250	8,626	8,694
30%	THE	14012	24,660	25,823	26,103
50%	THE	23353	41,090	43,018	43,479
100%	THE	46706	82,379	86,085	86,994
1%	FTX	468	858	866	869
10%	FTX	4671	8,634	8,734	8,755
30%	FTX	14012	25,827	26,191	26,276
50%	FTX	23353	43,052	43,632	43,763
100%	FTX	46706	86,192	87,326	87,581
1%	CLM	468	858	865	869
10%	CLM	4671	8,643	8,730	8,750
30%	CLM	14012	25,856	26,176	26,249
50%	CLM	23353	43,090	43,606	43,715
100%	CLM	46706	86,267	87,273	87,484

Table 6.4: Evaluation results after training on data augmented with Synonym Replacement in terms of micro F1-score with either OpenThesaurus (THE), fastText embeddings (FTX), or a contextual language model (CLM) as replacement source and different replacement percentages

		BiLSTM-CRF				XLM-R			
Dataset	Source	Baseline	20%	40%	60%	Baseline	20%	40%	60%
1%	THE	0.6994	-0.0035	**+0.0108**	+0.0096	0.6089	+0.0081	-0.0028	**+0.0322**
10%	THE	0.8941	**+0.0130**	+0.0073	+0.0042	0.9130	+0.0079	+0.0054	**+0.0104**
30%	THE	0.9346	+0.0010	+0.0012	**+0.0014**	0.9416	+0.0061	**+0.0097**	+0.0084
50%	THE	0.9430	**+0.0059**	+0.0033	+0.0040	0.9559	-0.0007	+0.0021	**+0.0031**
100%	THE	0.9572	+0.0023	**+0.0031**	+0.0013	0.9661	+0.0007	**+0.0010**	+0.0004
Ø			+0.0037	**+0.0051**	+0.0041		+0.0044	+0.0031	**+0.0109**
1%	FTX	0.6994	-0.0180	-0.0030	**+0.0072**	0.6089	**+0.0296**	+0.0224	+0.0268
10%	FTX	0.8941	+0.0047	+0.0073	**+0.0079**	0.9130	+0.0033	**+0.0055**	+0.0051
30%	FTX	0.9346	+0.0009	+0.0009	**+0.0023**	0.9416	**+0.0089**	+0.0063	+0.0075
50%	FTX	0.9430	**+0.0053**	+0.0048	+0.0049	0.9559	+0.0012	+0.0001	**+0.0022**
100%	FTX	0.9572	+0.0005	**+0.0025**	+0.0023	0.9661	+0.0011	**+0.0014**	+0.0002
Ø			-0.0013	+0.0025	**+0.0049**		**+0.0088**	+0.0071	+0.0084
1%	CLM	0.6994	**+0.0180**	-0.0039	+0.0007	0.6089	**+0.0943**	+0.0770	+0.0530
10%	CLM	0.8941	+0.0015	**+0.0089**	-0.0018	0.9130	**+0.0074**	+0.0018	+0.0013
30%	CLM	0.9346	+0.0002	**+0.0008**	+0.0002	0.9416	+0.0045	**+0.0047**	+0.0028
50%	CLM	0.9430	**+0.0047**	**+0.0047**	+0.0035	0.9559	**+0.0020**	-0.0004	-0.0005
100%	CLM	0.9572	**+0.0006**	-0.0005	-0.0018	0.9661	-0.0015	**-0.0003**	-0.0013
Ø			**+0.0050**	+0.0020	+0.0002		**+0.0213**	+0.0166	+0.0111

6.2.1.1 OpenThesaurus

Model Performance Using the OpenThesaurus database as the source of replacements, we achieve a maximum relative improvement in the BiLSTM-CRF model performance of 1.54% by augmenting the 1%-dataset with a replacement percentage of 40%. The maximum relative gain in the XLM-R model performance is 5.29% and is achieved by augmenting the 1% dataset with a replacement percentage of 60%. We notice that for the 1%- and 30%-dataset, the XLM-R model benefits more, while for the 10% dataset, the BiLSTM-CRF model has more considerable improvements. Generally, it seems that the XLM-R model trained on the larger datasets (50% and 100%) benefits only marginally from applying DA, while for the BiLSTM-CRF model, we get a mixed picture. It appears that with OpenThesaurus, a higher replacement percentage improves XLM-R performance, while for the BiLSTM-CRF model, it does not. We also notice one positive (+0.0130) and one negative (−0.0035) outlier in the data, namely the evaluation results after training on the 1%- and 10%-dataset augmented with a replacement percentage of 20% in combination with the BiLSTM-CRF model. Another example where the augmentation unexpectedly worsened the performance is the evaluation of the 1%-dataset augmented with a replacement percentage of 40% in combination with the XLM-R model.

Quality of Augmentation The quality of the synthetic sentences generated by applying Synonym Replacement using OpenThesaurus varies. There are cases where the retrieved replacements fit well into the sentence and others where the resulting sentence is barely comprehensible (cf. Table 6.5). Due to the limited information available in the OpenThesaurus database, the retrieved replacement does not always have the same meaning as the original token. A possible solution to this problem is the disambiguation of the original token and the retrieved candidates. This is a limitation of the chosen replacement source. Consequently, the generated synthetic sentences feature frequent syntactical errors and changes in semantics. For the English language, the WordNet thesaurus alleviates this problem by associating words with their POS tags [Sch94], which can then be used to filter the replacement candidates. Therefore, one direction for further research would be using GermaNET [HF97], which is a similar thesaurus for the German language, instead of OpenThesaurus. The upside of using OpenThesaurus is that the retrieved replacements are consistently sufficiently different from the original token. Surprisingly, the issues mentioned above do not significantly impact the model training negatively, and the added information from the synthetic sentences still causes improvements in model performance.

Table 6.5: Samples of sentences augmented with Synonym Replacement using OpenThesaurus as the source and a replacement-percentage of 40% with points of interest highlighted in blue and entities in italic

original	Die arbeitsvertraglichen Pflichten von Herrn *N* waren vielmehr für die Dauer des nach *§ 27 Abs. 2 MTV-DP AG* bewilligten Sonderurlaubs suspendiert .
augmented	Die arbeitsvertraglichen Pflichten vonseiten Herrn *N* waren eher für diese Dauer des nach *§ 27 Abs. 2 MTV-DP AG* bewilligten Sonderurlaubs suspendiert .
original	Der Beschwerdeführer habe sich in der Gruppe befunden , aus welcher es zu Provokationen und Körperverletzungsdelikten gekommen sei .
augmented	Der Ankläger Besitz sich in der Gesellschaftsschicht befunden , leer welcher es zu Provokationen auch Körperverletzungsdelikten gekommen sei .

6.2.1.2 fastText

Model Performance When retrieving replacements using the fastText embeddings, we achieve a maximum relative improvement in the BiLSTM-CRF model performance of 1.03% by augmenting the 1% dataset with a replacement percentage of 60%. The maximum relative gain in the XLM-R model performance is 4.86% and is achieved by augmenting the 1% dataset with a replacement percentage of 20%. Again, we notice that for the 1% and 30% datasets, the XLM-R model benefits more, while for the 10% dataset, the BiLSTM-CRF model shows greater overall improvement. It also appears that with fastText embeddings, a higher replacement percentage improves BiLSTM-CRF performance, while for the XLM-R model, it does not. Additionally, we notice that the performance of the XLM-R model after training on the larger datasets (50%, 100%) is impacted only very slightly by performing DA. In contrast, the BiLSTM-CRF model still shows improvements for the 50%-dataset. As before, we notice some deterioration in our data, namely the evaluation results after training on the 1% dataset augmented with a replacement percentage of 20% and 40% in combination with the BiLSTM-CRF model.

Quality of Augmentation The quality of the synthetic sentences generated by applying Synonym Replacement using fastText embeddings varies less than when using OpenThesaurus as the replacement source. In most cases, the meaning of retrieved replacements corresponds to the meaning of the original token. This can probably be attributed to the data structure of the embeddings and the used measure for calculating the similarity between embeddings (cosine similarity). Consequently, substituting tokens using fastText embeddings mostly leaves the semantics intact and only slightly degrades the syntax. The downside of this approach is that the retrieved replacements sometimes differ only in minimal ways, when, i.e., singular nouns are replaced with their plural forms, or correctly written words are replaced by their

misspelled versions (cf. Table 6.6). This phenomenon can be explained by the fact that the fastText embeddings were created from user-generated Twitter data, so the most similar available token often is the same word with a different spelling or declension.

Table 6.6: Samples of sentences augmented with Synonym Replacement using fastText embeddings as the source and a replacement-percentage of 40% with points of interest highlighted in blue and entities in italic

original	Beiden Rechtsmitteln bleibt der Erfolg versagt .
augmented	Beide Rechtsmitteln bliebt der Erfolg versagt .
original	Der Zug war nicht in *W.* , sondern in *X.* stationiert .
augmented	Der Zug ist nicht in *W.* , sondern in *X.* stationierte .
original	Das Anmeldezeichen benennt folglich den typischen Nutzer und die Funktion der gespeicherten Computerprogramme .
augmented	Das Anmeldezeichen benennt folglich diesen typischen Nutzerinnen sowie diese Funktion der gespeicherten Computerprogrammen .

6.2.1.3 Contextual Language Model

Model Performance Using the contextual language model XLM-RoBERTa to replace tokens, we achieve a maximum relative improvement in the BiLSTM-CRF model performance of 2.57% by augmenting the 1% dataset with a replacement percentage of 20%. The maximum relative gain in the XLM-R model performance is 15.49% and is achieved by augmenting the 1% dataset with a replacement percentage of 20%. Here, for the 1%, 10%, and 30% datasets, the XLM-R model benefits more from the augmentation than the BiLSTM-CRF model. In contrast to the other replacement sources, we notice that a higher replacement percentage does not increase but reduce the augmentation's positive impact across almost all dataset and model combinations. Again, the augmentation affects the performance of the models on the 100%-dataset only marginally. We notice that the augmentation unexpectedly worsened performance on the 1% dataset augmented with a replacement percentage of 40% using the BiLSTM-CRF model.

Quality of Augmentation The quality of the synthetic sentences generated by applying Synonym Replacement using the contextual language model XLM-RoBERTa is similar to the quality of the sentences generated using fastText embeddings. The semantics of the sentences change more than with the other replacement sources, while the syntax remains intact in most cases (cf. Table 6.7 sample A). In contrast to the fastText embeddings, the contextual language model rarely returns replacement candidates that differ only slightly from the original token. In some cases, the replacement does not have the same meaning as the original token but

is a different token. This replacement source seems to be a compromise between the more accurate replacements returned by the fastText embeddings and the higher probability of introducing differences compared to the original sentence when using the OpenThesaurus as the replacement source. When examining the sentences augmented with higher replacement percentages (40%, 60%), we found that replacing too many original tokens in a sentence with replacements suggested by a contextual language model occasionally strongly alters the original semantics (cf. Table 6.7 sample B).

Table 6.7: Samples of sentences augmented with Synonym Replacement using a contextual language model as the source and a replacement-percentage of 40% with points of interest highlighted in blue and entities in italic

A	**original**	Der Beschwerdeführer habe sich in der Gruppe befunden , aus welcher es zu Provokationen und Körperverletzungsdelikten gekommen sei .
	augmented	Die Beschwerdeführer habe sich vor einer Situation gefunden , aus welcher heraus die Provokationen zu Körperverletzungsdelikten geworden ist .
B	**original**	Zwar gelten für die Inanspruchnahme des Gläubigers der Kapitalerträge als Schuldner der Kapitalertragsteuer andere Rechtsgrundlagen als für den Schuldner der Kapitalerträge , der die Steuer für Rechnung des Steuerschuldners zu entrichten hat (*§ 43 Satz 2 AO*) .
	augmented	Es gelten über die Haftung des Gläubigers der Beiträge vom Schuldner zur Kapitalertragsteuer andere Vorschriften wie für einen Schuldner ohne Anspruch, der diese jedoch auf Kosten des Unternehmens zur entrichten hatte (*§ 43 Satz 2 AO*)

6.2.1.4 Conclusion

Figure 6.1 depicts a summary of the results from evaluating different configurations of Synonym Replacement with regard to replacement source and percentage. It displays the relative improvement achieved after using DA compared to the baseline value for the corresponding dataset and model combination. Note that the scale on the y-axis is logarithmic for values larger than ±1%. In most cases applying Synonym Replacement leads to performance improvements. We notice that, especially with the XLM-R model, the improvement for the two largest datasets is minimal. From Table 6.4 we deduce that the contextual language model as source is best used with a low replacement percentage; for the other sources, we get mixed results. The maximum improvement across the different replacement sources for the BiLSTM-CRF model and the XLM-R model was achieved by augmenting the 1%-dataset with a replacement percentage of 20% using the contextual language model as the replacement source, with gains of 2.57% and 15.49% respectively. Figure 6.2 displays the average relative improvement in micro F1-score across all datasets achieved by applying Synonym Replacement depending on replacement source and percentage with both models. Generally, the XLM-R model benefits

more from DA than the BiLSTM-CRF model, with the contextual language model yielding the greatest average improvement. However, a big share of the average improvement can be attributed to the gain on the 1%-dataset, and it should be taken into account that the baseline performance on this dataset is considerably lower for the XLM-R model than for the BiLSTM-CRF model. The main issue with this technique is that it is relatively time consuming, depending on the used configuration, although it might be possible to speed up the process significantly by using parallelization.

6.2.2 Mention Replacement

The Mention Replacement augmentation technique is the least expensive of the three tested techniques. Augmenting the 100%-dataset only takes 499 seconds, corresponding to only 0.011 seconds per sentence. By applying Mention Replacement we were able to augment 17,678 of the 46,706 original sentences, increasing the dataset size by 37.85% (cf. Table 6.8). This number is relatively low as 24,774 (53.04%) sentences did not contain entities and consequently were not affected by Mention Replacement. In addition, the sentences with irreproducible tokenizations were not eligible for augmentation, which accounts for another 9.06% of the original dataset. Besides that, there were 21 sentences that did not change even though they contained at least one entity. This is probably due to these entities coincidentally being replaced by themselves, which is likely to happen with classes with very few entities. In total, 29,949 entities across all classes were swapped during the augmentation.

Model Performance Table 6.9 lists the results we achieved after training both models on the augmented datasets. The BiLSTM-CRF model benefits from the augmented data across the entire range of dataset sizes, with improvements of up to 3.17%. The XLM-R model shows more significant performance improvements for the small datasets but lower improvements for the larger 50%- and 100%-dataset with gains of up to 12.68%. The 100%-dataset is the only instance where the XLM-R model performance worsened slightly compared to the baseline, but the improvements for all settings with datasets larger than the 10%-dataset are minor. The absolute and relative maximum improvement for both models is achieved for the 1%-dataset, the absolute and relative minimum for the 100%-dataset. The average change in micro F_1-score across all datasets is +0.0075 for the BiLSTM-CRF model and +0.0194 for the XLM-R model.

Quality of Augmentation The quality of the synthetic sentences generated by applying the Mention Replacement technique is consistent and overall very good. While the syntax of the sentences does not change significantly, this also assures that no significant distortions or grammatical mistakes are introduced. Most of the few errors we encountered during the

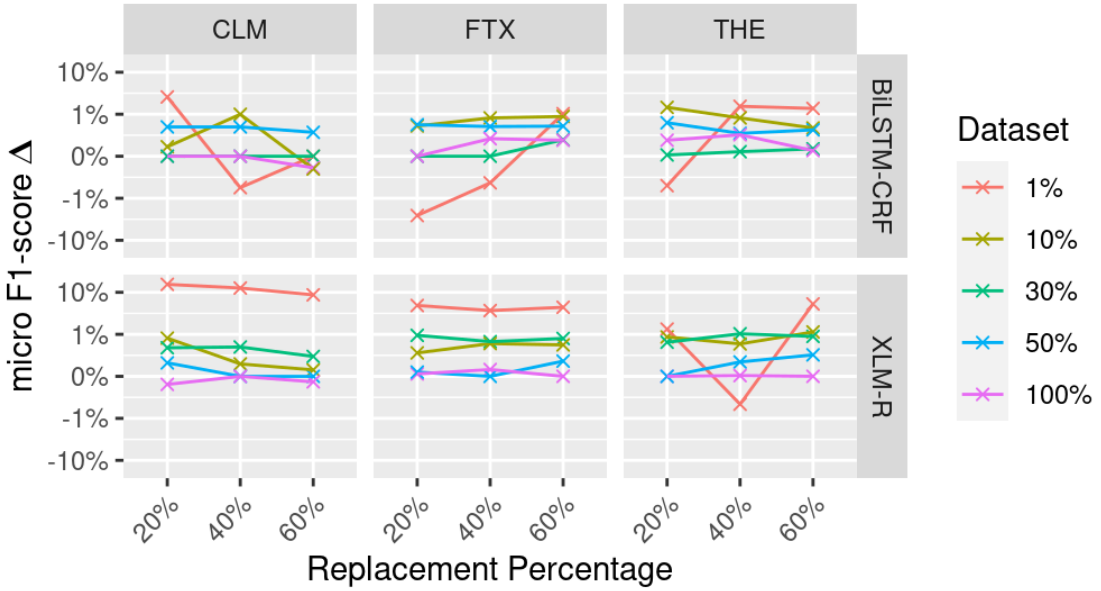


Figure 6.1: Micro F1-score improvements achieved by applying Synonym Replacement over replacement percentage by source, model and dataset

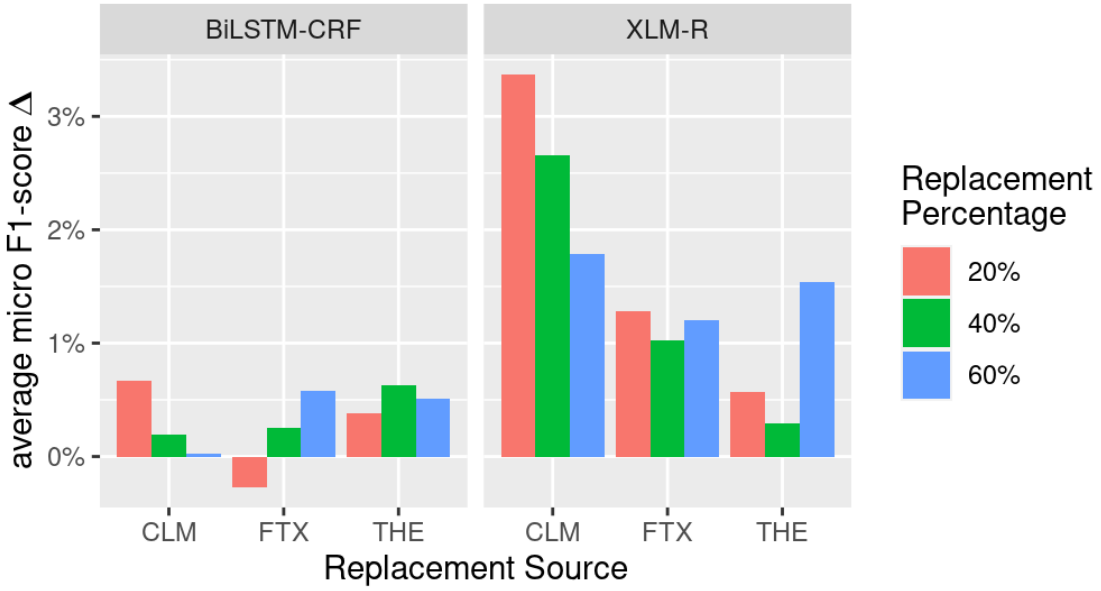


Figure 6.2: Average micro F1-score improvements across all datasets achieved by applying Synonym Replacement by source, model and percentage

manual evaluation were issues related to the improper use of articles or similar errors caused by a change in subject or object in a sentence (cf. Table 6.10). In contrast to, e.g., Synonym Replacement, Mention Replacement strongly influences the semantics of sentences, as objects and subjects are randomly exchanged. This, however, does not seem to affect the models negatively. A reasonable explanation is that the models do not develop an understanding of the sentence and word semantics but rather learn to distinguish the classes based on the entire sentence structure and composition.

Mention Replacement proves an effective DA technique that can add value for relatively low time and computing cost. An interesting direction to explore is how effectively Mention Replacement might be used for upsampling [Prooo], in which the class distribution of the training data is artificially adjusted to be more balanced.

6.2.3 Back-Translation

The Back-Translation augmentation technique is not very resource expensive but still, due to API rate limits associated with the `BackTranslation` package, augmenting the 100%-dataset took 154 hours, corresponding to 11.87 seconds per sentence. By applying Back-Translation we were able to augment 29, 594 of the 46, 706 sentences, increasing the dataset size by 63.24% (cf. Table 6.8). This initially seems like a good result, but out of these 29, 594 sentences, only 5, 449 sentences contain at least one or more entities. The majority of generated sentences do not contain entities. In addition, the sentences with irreproducible tokenizations were not eligible for augmentation, which accounts for another 9.06% of the original dataset. There were 457 sentences that did not change despite being back-translated. This can probably be attributed to very short or simple sentences not leaving much room for changes.

Model Performance Table 6.9 lists the results we achieved after training both models on the augmented datasets. We do not register a significant impact of the Back-Translation augmentation on the performance regarding the micro F1-score of either the BiLSTM-CRF or the XLM-R model. However, it seems that in most cases, it slightly deteriorates the performance of the XLM-R model. The average change in micro F1-score across all datasets is +0.0018 for the BiLSTM-CRF model and −0.0025 for the XLM-R model.

Quality of Augmentation The quality of the synthetic sentences generated by applying the Back-Translation technique varies substantially. Short sentences with and without entities are mostly translated well and show moderate rates of change compared to the original sentence (cf. Table 6.11 sample A, B). Longer sentences and sentences containing special terminology from the legal domain, on the other hand, occasionally suffer from a loss of meaning and

Table 6.8: Dataset sizes after applying Mention Replacement, Back-Translation

Dataset	Original	MR	BT
1%	468	641	744
10%	4671	6,420	7,648
30%	14012	19,284	22,899
50%	23353	32,120	38,141
100%	46706	64,384	76,242

Table 6.9: Evaluation results after training on data augmented with Mention Replacement (MR) or Back-Translation (BT) in terms of micro F1-score

	BiLSTM-CRF			XLM-R		
Dataset	**Baseline**	**MR Δ**	**BT Δ**	**Baseline**	**MR Δ**	**BT Δ**
1%	0.6994	**+0.0222**	+0.0065	0.6089	**+0.0772**	-0.0123
10%	0.8941	**+0.0053**	-0.0040	0.9130	**+0.0103**	-0.0025
30%	0.9346	**+0.0032**	+0.0006	0.9416	**+0.0064**	+0.0037
50%	0.9430	**+0.0061**	+0.0063	0.9559	**+0.0033**	-0.0008
100%	0.9572	**+0.0007**	-0.0003	0.9661	-0.0003	**-0.0004**
Ø		+0.0075	+0.0018		+0.0194	-0.0025

Table 6.10: Samples of sentences augmented with Mention Replacement with points of interest highlighted in blue and entities in italic

A	**original**	(2) Ob die Umwandlung der Todesstrafe in eine lebenslange Freiheitsstrafe bereits zwingend aus dem seit 1991 praktizierten Moratorium folgt , wie es das *Bundesverwaltungsgericht* angenommen hat , kann dahinstehen .
	augmented	(2) Ob die Umwandlung der Todesstrafe in eine lebenslange Freiheitsstrafe bereits zwingend aus dem seit 1991 praktizierten Moratorium folgt , wie es das *Europäischen Gerichtshof* angenommen hat , kann dahinstehen .
B	**original**	Eine gesetzliche Verpflichtung zur Mitgliedschaft in der berufsständischen Kammer bestand bereits vor dem 1. 1. 1995 in *Baden-Württemberg* .
	augmented	Eine gesetzliche Verpflichtung zur Mitgliedschaft in der berufsständischen Kammer bestand bereits vor dem 1. 1. 1995 in *Türkei* .

inaccuracies (cf. Table 6.11 sample C, D). We assume that the main cause for such semantic issues is the translation service, which sometimes returns poor translations (cf. Table 6.11 sample D). We assume that the root of this issue partly is that not all concepts of the German legal system have equivalents in the legal systems of the country whose language is used as pivotal language, in our case English. Another big challenge is the re-annotation of the entities in the back-translated sentences. This currently only works in approximately every fourth sentence, leading to most (successfully) augmented sentences not containing any entities. Only 5, 449 sentences containing one or more entities were augmented successfully. In contrast, in the original dataset, 21, 175 sentences contained one or more entities. The total number of annotated entities increased by 17.52% (cf. Table 6.12). Note that the classes Court decision (RS), Legal literature (LIT), EU legal norm (EUN), and Ordinance (VO) have particularly low augmentation success rates. The re-annotation of entities is impeded by the quality of the translations, as they sometimes alter the entities in unexpected and inconsistent ways causing the back-translated entities as well as the original entity to be different compared to the entities that were back-translated as part of the sentence. In sample E of Table 6.11, the back-translated sentence contains a mixture of the original entity (”§ 173 Abs. 1 Nr. 2”) and the back-translated entity (”Steuergesetzbuch”). Moreover, the article and the declension of the object match neither the original nor the back-translated entity. These issues might be alleviated using another translation service such as Googles Cloud Translation or DeepL, as small-scale tests show that the translations are of higher quality, or Another parameter that might influence the quality of translations is the pivotal language, although we did not perform tests using other pivotal languages.

Despite the issues mentioned above, the sentences' syntax is often not affected as severely as the semantics. In the German LER dataset, a significant share of all sentences is long and complex. After augmentation, these sentences still have correct syntax, but their original meaning is frequently lost. Concerning the differences between augmented and original sentences, we observe that in many cases, only the wording changes, but the structure of the sentence remains unchanged. This is not what we were hoping for, as Back-Translation is expected to paraphrase the original sentence while retaining its meaning (cf. Section 4.3). The lack of change in structure might be related to the structure of the input sentences, their length, the chosen pivotal language, or the chosen translation service.

We conclude that Back-Translation might be better suited to domains that do not come with long, elaborate sentence structures and language-dependent terminology. Then the re-annotation of entities is more likely to succeed as the entities will be less complex, and the translation results can be expected to be of consistently higher quality. In the current state, it is not a suitable augmentation technique for NER over the LER dataset.

Table 6.11: Samples of sentences augmented with Back-Translation with points of interest highlighted in blue and entities in italic. Sentences with the remark "failed" were not successfully re-annotated and therefore not added to the augmented dataset.

A	**original**	Auch die Zahlung von Teilbeträgen sei nicht zu erwarten gewesen .
	augmented	Die Zahlung von Teilbeträgen war ebenfalls nicht zu erwarten .
B	**original**	Der Zug war nicht in *W.* , sondern in *X.* stationiert .
	augmented	Der Zug war nicht in *W.* stationiert , sondern in *X.*
C	**original**	Der Kläger hat zuletzt - soweit für das Revisionsverfahren von Interesse - beantragt ,
	augmented	Der Kläger hat kürzlich Zinsen beantragt - bis zum Revisionsverfahren
D	**original**	Auch wenn das Schätzungsergebnis trotz vorhandener Möglichkeiten , den Sachverhalt aufzuklären und Schätzungsgrundlagen zu ermitteln , von den tatsächlichen Gegebenheiten abweicht und in keiner Weise erkennbar ist , dass überhaupt und ggf. welche Schätzungserwägungen angestellt wurden (s. *BFH-Urteil / NV 2015 , 145* , m. w. N.) , kann eine Nichtigkeit vorliegen .
	augmented (failed)	Auch wenn das Ergebnis der Schätzung trotz der Möglichkeiten , die Tatsachen zu klären und die Bewertungen zu ermitteln */ NV 2015 , 145* , m. W. N.) , kann es Nichtigkeit geben .
E	**original**	Im Jahr 2012 beantragte der Kläger die Berücksichtigung der Versorgungsleistungen in den Streitjahren und begehrte die Änderung dieser Einkommensteuerfestsetzungen nach *§ 173 Abs. 1 Nr. 2 der Abgabenordnung* .
	back-translated entity	Abschnitt 173 (1) Nr. 2 des Steuergesetzbuchs
	augmented (failed)	Im Jahr 2012 beantragte der Kläger die Rentenleistungen in den Streitjahren und beantragte die Änderung dieser Einkommensteuerbewertungen gemäß *§ 173 Abs. 1 Nr. 2 der Steuergesetzbuch* .

Table 6.12: Number of annotated entities per class before and after applying Back-Translation

Class	Description	Original	Augmented	Δ
PER	Person	1,240	1,835	+47.98%
ST	City	475	702	+47.79%
AN	Lawyer	80	116	+45.00%
LD	Country	1,006	1,450	+44.14%
LDS	Landscape	138	198	+43.48%
MRK	Brand	202	284	+40.59%
UN	Company	745	1,020	+36.91%
STR	Street	101	138	+36.63%
RR	Judge	1,052	1,342	+27.57%
ORG	Organization	823	1,025	+24.54%
GRT	Court	2,254	2,748	+21.92%
INN	Institution	1,534	1,852	+20.73%
VT	Contract	2,034	2,401	+18.04%
GS	Law	12,890	15,068	+16.90%
VS	Regulation	434	501	+15.44%
VO	Ordinance	559	627	+12.16%
RS	Court decision	8,839	9,536	+7.89%
LIT	Legal literature	2,115	2,220	+4.96%
EUN	EU legal norm	1,034	1,073	+3.77%
Total		**37,555**	**44,136**	**+17.52%**

7 Conclusion and Future Work

The goal of this work was to first select machine learning models for the NER task, and the subsequent evaluation of DA techniques, examine how well DA techniques for the NER task work with a specific focus on the German legal domain, how different (German) sources of replacements for Synonym Replacement compare to each other, and to explore a new variant of applying Back-Translation to sequence-labeling data. To answer these questions, we implemented and evaluated DA techniques. This included surveying the DA techniques applied to sequence-labeling data, selecting two different state-of-the-art DL models, implementing three DA techniques, and evaluating their effect on the selected models' performance.

We selected two model architectures, BiLSTM-CRF and a fine-tuned Transformer (XLM-RoBERTa), that are frequently used in the community and reflect the current state-of-the-art. Figure 7.1 displays the relative improvements achieved in model performance after applying the DA techniques to the datasets. Due to the different configurations tested, we have multiple data points per dataset and model for the Synonym Replacement technique. We notice that applying DA can be very beneficial when working with small datasets such as the 1%-dataset that contains only 468 sentences. Mention Replacement offers good value considering its low computational complexity compared to Synonym Replacement and Back-Translation. The impact of using Mention Replacement and Synonym Replacement is comparable. However, for the smallest dataset, Mention Replacement works slightly better with the BiLSTM-CRF model and slightly worse with the XLM-R model than Synonym Replacement. While applying Synonym and Mention Replacement led to improvements in model performance, we found that Back-Translation might be better suited to other domains. This is, among other things, due to the long and nested sentences encountered in the legal domain and the fact that not all concepts of the German legal system have equivalents in the other legal systems, making translation difficult. It does not yield good results, often even deteriorating performance. On the other hand, Synonym Replacement improved model performance in most cases, mainly when applied to the smaller 1%- and 10%-dataset. Concerning our initial goals, we achieved all of them, namely the implementation of Synonym Replacement, Mention Replacement and

Back-Translation, as well as their evaluation, including the comparison of different replacement sources and percentages for Synonym Replacement. Other investigation directions that we would have liked to include but were unable to due to time constraints are evaluating multiple different pivotal languages during Back-Translation and assessing the combined application of the implemented DA techniques. We could not run the evaluations multiple times to report the mean metrics and their standard deviation.

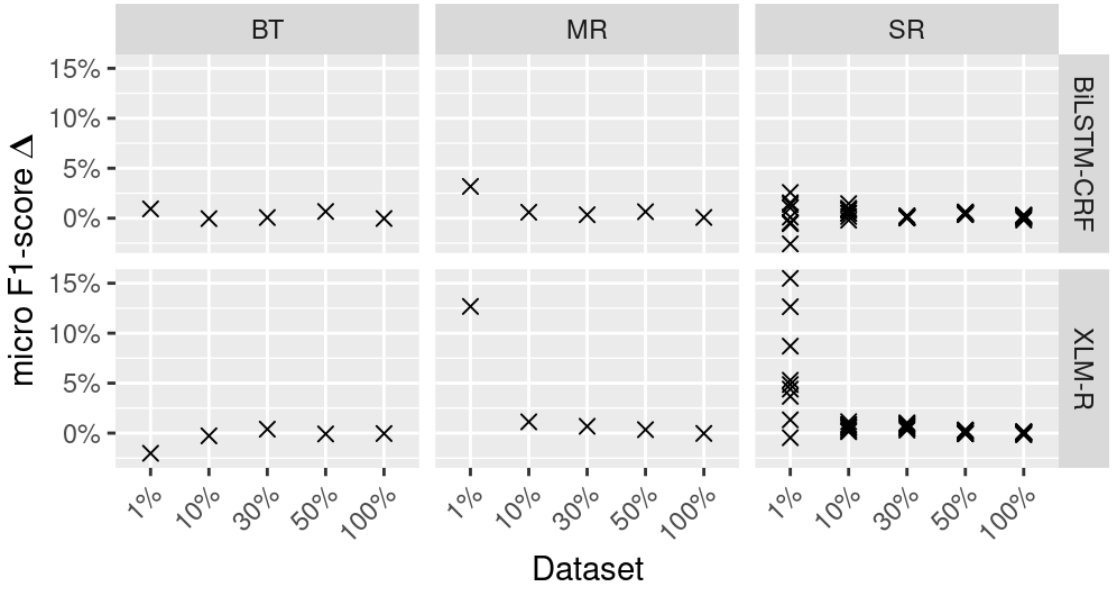


Figure 7.1: Micro F1-score improvements achieved by applying Back-Translation (BT), Mention Replacement (MR) and Synonym Replacement (SR) by model and dataset

The main issues that we faced occurred during implementation and evaluation. The python package we used for translation was not stable, requiring us to implement a checkpointing system. It had to restart from the last checkpoint 71 times during the entire augmentation process. Besides that, only 5, 449 sentences that were augmented contained entities, in contrast to 21, 175 sentences in the original dataset containing one or more entities, meaning augmentation frequently failed for sentences containing entities. OpenThesaurus was returning worse replacements than we anticipated, i.e., suggesting "marrow" as a replacement for "this", and the augmentation with Synonym Replacement and Back-Translation took longer than initially expected, expanding our targeted completion time frame. In addition, the repeated evaluation of the augmented datasets was also a very timely procedure.

Possible future research directions in this area are further improving the Back-Translation technique by, e.g., using other translation services, refining the re-annotation process of entities in the back-translated sentence, and testing other pivotal languages. All implemented DA techniques might substantially benefit from optimizations regarding their performance, i.e., parallelization, making their usage less expensive and facilitating further research. Mention

Replacement could be extended to optionally include a manually composed dictionary, or a dictionary gathered from knowledge graphs to introduce new entities into the dataset. The performance of Synonym Replacement when used with a thesaurus as the replacement source could be improved by using a thesaurus that supports filtering words by their POS tag. One such source that may be explored is GermaNET [HF97]. The retrieval of replacements from fastText embeddings could possibly be improved by requiring a minimum edit distance relative to the original token.

References

[ABV18] A Akbik, D Blythe, and R Vollgraf. Contextual String Embeddings for Sequence Labeling. In: *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*. Ed. by EM Bender, L Derczynski, and P Isabelle. Association for Computational Linguistics, 2018, pp. 1638–1649.

[Akb+19] A Akbik et al. FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*. Ed. by W Ammar, A Louis, and N Mostafazadeh. Association for Computational Linguistics, 2019, pp. 54–59. DOI: 10.18653/v1/n19-4010.

[AZS06] DM Aliod, M van Zaanen, and D Smith. Named Entity Recognition for Question Answering. In: *Proceedings of the Australasian Language Technology Workshop, ALTA 2006, Sydney, Australia, November 30-December 1, 2006*. Ed. by L Cavedon and I Zukerman. Australasian Language Technology Association, 2006, pp. 51–58.

[BH03] B Babych and A Hartley. Improving Machine Translation Quality with Automatic Named Entity Recognition. In: *Proceedings of the 7th International EAMT workshop on MT and other language technology tools, Improving MT through other language technology tools, Resource and tools for building MT at EACL 2003*. 2003. https://aclanthology.org/W03-2201.

[Bad18] I Badji. Legal Entity Extraction with NER Systems. PhD thesis. ETSI_Informatica, 2018.

[Bow+15] SR Bowman et al. A large annotated corpus for learning natural language inference. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 632–642. DOI: 10.18653/v1/D15-1075.

[Bru18] S Brugman. Deep Learning for Legal Tech: exploring NER on Dutch court rulings. PhD thesis. Radboud University, 2018.

[CSM20] B Chan, S Schweter, and T Möller. German's Next Language Model. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 6788–6796. DOI: 10.18653/v1/2020.coling-main.598.

[Cha+14] K Chatfield et al. Return of the Devil in the Details: Delving Deep into Convolutional Nets. In: *British Machine Vision Conference, BMVC 2014, Nottingham, UK, September 1-5, 2014*. Ed. by MF Valstar, AP French, and TP Pridmore. BMVA Press, 2014.

[Con+20] A Conneau et al. Unsupervised Cross-lingual Representation Learning at Scale. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747.

[DA20] X Dai and H Adel. An Analysis of Simple Data Augmentation for Named Entity Recognition. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 3861–3867. DOI: 10.18653/v1/2020.coling-main.343.

[Der+17] L Derczynski et al. Results of the WNUT2017 Shared Task on Novel and Emerging Entity Recognition. In: *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 140–147. DOI: 10.18653/v1/W17-4418.

[Dev+19] J Devlin et al. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by J Burstein, C Doran, and T Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/n19-1423.

[Din+20] B Ding et al. DAGA: Data Augmentation with a Generation Approach for Low-resource Tagging Tasks. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 6045–6057. DOI: 10.18653/v1/2020.emnlp-main.488.

[Dod+04] GR Doddington et al. The Automatic Content Extraction (ACE) Program - Tasks, Data, and Evaluation. In: *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*. European Language Resources Association, 2004.

[DLL14] RI Dogan, R Leaman, and Z Lu. NCBI disease corpus: A resource for disease name recognition and concept normalization. In: *J. Biomed. Informatics* 47:(2014), 1–10. DOI: 10.1016/j.jbi.2013.12.006.

[Elm90] JL Elman. Finding Structure in Time. In: *Cogn. Sci.* 14(2):(1990), 179–211. DOI: 10.1207/s15516709cog1402_1.

[Etz+05] O Etzioni et al. Unsupervised named-entity extraction from the Web: An experimental study. In: *Artificial Intelligence* 165(1):(2005), 91–134. ISSN: 0004-3702. DOI: 10.1016/j.artint.2005.03.001.

[GG16] Y Gal and Z Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain:

Curran Associates Inc., 2016, pp. 1027–1035. ISBN: 9781510838819.

[GSC99] F Gers, J Schmidhuber, and F Cummins. Learning to forget: continual prediction with LSTM. In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. Vol. 2. 1999, 850–855 vol.2. DOI: 10.1049/cp:19991218.

[GWM18] I Glaser, B Waltl, and F Matthes. Named entity recognition, extraction, and linking in German legal contracts. In: *IRIS: Internationales Rechtsinformatik Symposium*. 2018, pp. 325–334.

[GGK18] A Goyal, V Gupta, and M Kumar. Recent Named Entity Recognition and Classification techniques: A systematic review. In: *Computer Science Review* 29:(2018), 21–43. ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2018.06.001.

[GBV20] M Grandini, E Bagli, and G Visani. Metrics for Multi-Class Classification: an Overview. In: *CoRR* abs/2008.05756:(2020).

[Gra+18] E Grave et al. Learning Word Vectors for 157 Languages. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. Ed. by N Calzolari et al. European Language Resources Association (ELRA), 2018.

[GS96] R Grishman and B Sundheim. Message Understanding Conference-6: A Brief History. In: *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*. COLING '96. Copenhagen, Denmark: Association for Computational Linguistics, 1996, pp. 466–471. DOI: 10.3115/992628.992709.

[HF97] B Hamp and H Feldweg. GermaNet - a Lexical-Semantic Net for German. In: *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*. 1997.

[HS97] S Hochreiter and J Schmidhuber. Long Short-Term Memory. In: *Neural Comput.* 9(8):(1997), 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[HXY15] Z Huang, W Xu, and K Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. In: *CoRR* abs/1508.01991:(2015).

[IG21] AM Issifu and MC Ganiz. A Simple Data Augmentation Method to Improve the Performance of Named Entity Recognition Models in Medical Domain. In: *2021 6th International Conference on Computer Science and Engineering (UBMK)*. 2021, pp. 763–768. DOI: 10.1109/UBMK52708.2021.9558986.

[JK15] H Jabbar and RZ Khan. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). In: *Computer Science, Communication and Instrumentation Devices* 70:(2015).

[Kan+20] T Kang et al. UMLS-based data augmentation for natural language processing of clinical research literature. In: *Journal of the American Medical Informatics Association* 28(4):(Dec. 2020), 812–823. ISSN: 1527-974X. DOI: 10.1093/jamia/ocaa309.

[KBC20] A Keraghel, K Benabdeslem, and B Canita. Data augmentation process to improve deep learning-based NER task in the automotive industry field. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207241.

[Ko+15] T Ko et al. Audio augmentation for speech recognition. In: *Proc. Interspeech 2015*. 2015, pp. 3586–3589. DOI: 10.21437/Interspeech.2015-711.

[KK15] M Konkol and M Konopík. Segment Representations in Named Entity Recognition. In: *Text, Speech, and Dialogue*. Ed. by P Král and V Matoušek. Cham: Springer International Publishing, 2015, pp. 61–70. ISBN: 978-3-319-24033-6.

[Kun+20] HK Kung et al. Data-Augmented Hybrid Named Entity Recognition for Disaster Management by Transfer Learning. In: *Applied Sciences* 10(12):(2020). ISSN: 2076-3417. DOI: 10.3390/app10124234.

[LMP01] JD Lafferty, A McCallum, and FCN Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*. Ed. by CE Brodley and AP Danyluk. Morgan Kaufmann, 2001, pp. 282–289.

[Lam+16] G Lample et al. Neural Architectures for Named Entity Recognition. 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, June 2016, pp. 260–270. DOI: 10.18653/v1/N16-1030.

[LRM19] E Leitner, G Rehm, and J Moreno-Schneider. Fine-Grained Named Entity Recognition in Legal Documents. In: *Semantic Systems. The Power of AI and Knowledge Graphs*. Ed. by M Acosta et al. Cham: Springer International Publishing, 2019, pp. 272–287. ISBN: 978-3-030-33220-4.

[LRS20] E Leitner, G Rehm, and JM Schneider. A Dataset of German Legal Documents for Named Entity Recognition. In: *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*. Ed. by N Calzolari et al. European Language Resources Association, 2020, pp. 4478–4485.

[Li+22] J Li et al. A Survey on Deep Learning for Named Entity Recognition. In: *IEEE Transactions on Knowledge and Data Engineering* 34(1):(2022), 50–70. DOI: 10.1109/TKDE.2020.2981314.

[Liu+20] Q Liu et al. Long-tail Dataset Entity Recognition based on Data Augmentation. In: *Proceedings of the 1st Workshop on Extraction and Evaluation of Knowledge Entities from Scientific Documents co-located with the ACM/IEEE Joint Conference on Digital Libraries in 2020, EEKE@JCDL 2020, Virtual Event, China, August 1st, 2020*. Ed. by C Zhang et al. Vol. 2658. CEUR Workshop Proceedings. CEUR-WS.org, 2020, pp. 79–80.

[Liu+19] Y Liu et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. In: *CoRR* abs/1907.11692:(2019).

[LH19] I Loshchilov and F Hutter. Decoupled Weight Decay Regularization. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[Luq19] FM Luque. Atalaya at TASS 2019: Data Augmentation and Robust Embeddings for Sentiment Analysis. In: *Proceedings of the Iberian Languages Evaluation Forum co-located with 35th Conference of the Spanish Society for Natural Language Processing, IberLEF@SEPLN 2019, Bilbao, Spain, September 24th, 2019*. Ed. by MÁG

Cumbreras et al. Vol. 2421. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 561–570.

[Mar+13] M Marrero et al. Named Entity Recognition: Fallacies, challenges and opportunities. In: *Computer Standards & Interfaces* 35(5):(2013), 482–489. ISSN: 0920-5489. DOI: https://doi.org/10.1016/j.csi.2012.09.004.

[Men+11] PN Mendes et al. DBpedia Spotlight: Shedding Light on the Web of Documents. In: *Proceedings of the 7th International Conference on Semantic Systems*. I-Semantics '11. Graz, Austria: Association for Computing Machinery, 2011, pp. 1–8. ISBN: 9781450306218. DOI: 10.1145/2063518.2063519.

[Mik+13] T Mikolov et al. Efficient Estimation of Word Representations in Vector Space. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Ed. by Y Bengio and Y LeCun. 2013.

[MT16] J Mueller and A Thyagarajan. Siamese Recurrent Architectures for Learning Sentence Similarity. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Ed. by D Schuurmans and MP Wellman. AAAI Press, 2016, pp. 2786–2792.

[Mys+19] S Mysore et al. The Materials Science Procedural Text Corpus: Annotating Materials Synthesis Procedures with Shallow Semantic Structures. In: *Proceedings of the 13th Linguistic Annotation Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 56–64. DOI: 10.18653/v1/W19-4007.

[Nab05] D Naber. OpenThesaurus: ein offenes deutsches Wortnetz. In: *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen: Beiträge zur GLDV-Tagung, Bonn, Germany*:(2005), 422–433.

[NS09] D Nadeau and S Sekine. A survey of named entity recognition and classification. In: *Benjamins Current Topics*. John Benjamins Publishing Company, 2009, pp. 3–28. DOI: 10.1075/bct.19.03nad.

[Nak+21] K Nakamura et al. Learning-Rate Annealing Methods for Deep Neural Networks. In: *Electronics* 10(16):(2021). ISSN: 2079-9292. DOI: 10.3390/electronics10162029.

[NJM21] Z Nasar, SW Jaffry, and MK Malik. Named Entity Recognition and Relation Extraction: State-of-the-Art. In: *ACM Comput. Surv.* 54(1):(Feb. 2021). ISSN: 0360-0300. DOI: 10.1145/3445965.

[Net19] P Netrapalli. Stochastic Gradient Descent and Its Variants in Machine Learning. In: *Journal of the Indian Institute of Science* 99(2):(2019), 201–213. DOI: 10.1007/s41745-019-0098-4.

[Nye+18] B Nye et al. A Corpus with Multi-Level Annotations of Patients, Interventions and Outcomes to Support Language Processing for Medical Literature. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 197–207. DOI: 10.18653/v1/P18-1019.

[Paf+13] E Pafilis et al. The SPECIES and ORGANISMS Resources for Fast and Accurate Identification of Taxonomic Names in Text. In: *PloS one* 8(6):(2013), e65390. DOI: 10.1371/journal.pone.0065390.

[PSM14] J Pennington, R Socher, and 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.

[Pet+17] ME Peters et al. Semi-supervised sequence tagging with bidirectional language models. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Ed. by R Barzilay and M Kan. Association for Computational Linguistics, 2017, pp. 1756–1765. DOI: 10.18653/v1/P17-1161.

[Pet+18] ME Peters et al. Deep Contextualized Word Representations. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202.

[PU16] T Proisl and P Uhrig. SoMaJo: State-of-the-art tokenization for German web and social media texts. In: *Proceedings of the 10th Web as Corpus Workshop*. Berlin: Association for Computational Linguistics, Aug. 2016, pp. 57–62. DOI: 10.18653/v1/W16-2607.

[Pro] G Project. GENIA Project - Corpus. https://web.archive.org/web/20220711150959/http://www.geniaproject.org/genia-corpus. [Online; accessed 11-July-2022].

[Pro00] F Provost. Machine learning from imbalanced data sets 101. In: *Proceedings of the AAAI'2000 workshop on imbalanced data sets*. Vol. 68. (2000). AAAI Press. 2000, pp. 1–3.

[RM17] J Raiman and J Miller. Globally Normalized Reader. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 1059–1069. DOI: 10.18653/v1/D17-1111.

[Raj+16] P Rajpurkar et al. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264.

[RG17] N Reimers and I Gurevych. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 338–348. DOI: 10.18653/v1/D17-1035.

[Ros61] F Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[RHW86] DE Rumelhart, GE Hinton, and RJ Williams. Learning representations by back-propagating errors. In: *Nature* 323(6088):(1986), 533–536. DOI: 10.1038/323533a0.

[Sab+21] C Sabty et al. Data Augmentation Techniques on Arabic Data for Named Entity Recognition. In: *Procedia Computer Science* 189:(2021). AI in Computational Linguistics, 292–299. ISSN: 1877-0509. DOI: 10.1016/j.procs.2021.05.092.

[SM03] EFTK Sang and FD Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In: *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*. Ed. by W Daelemans and M Osborne. ACL, 2003, pp. 142–147.

[SG15] CN dos Santos and V Guimarães. Boosting Named Entity Recognition with Neural Character Embeddings. In: *Proceedings of the Fifth Named Entity Workshop*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 25–33. DOI: 10.18653/v1/W15-3904.

[SZ14] CN dos Santos and B Zadrozny. Learning Character-level Representations for Part-of-Speech Tagging. In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. Vol. 32. JMLR Workshop and Conference Proceedings. JMLR.org, 2014, pp. 1818–1826.

[Sar08] S Sarawagi. *Information extraction*. Now Publishers Inc, 2008.

[Sch94] H Schmid. Part-Of-Speech Tagging With Neural Networks. In: *15th International Conference on Computational Linguistics, COLING 1994, Kyoto, Japan, August 5-9, 1994*. 1994, pp. 172–176.

[SA20] S Schweter and A Akbik. FLERT: Document-Level Features for Named Entity Recognition. In: *CoRR* abs/2011.06993:(2020).

[SHB16] R Sennrich, B Haddow, and A Birch. Improving Neural Machine Translation Models with Monolingual Data. 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. 86–96. DOI: 10.18653/v1/P16-1009.

[Sha+10] K Shaalan et al. Rule-based approach in Arabic natural language processing. In: *The International Journal on Information and Communication Technologies (IJICT)* 3(3):(June 2010), 11–19.

[Shi+20] H Shim et al. Data Augmentation and Semi-Supervised Learning for Deep Neural Networks-Based Text Classifier. In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1119–1126. ISBN: 9781450368667. DOI: 10.1145/3341105.3373992.

[Sri+14] N Srivastava et al. Dropout: a simple way to prevent neural networks from overfitting. In: *J. Mach. Learn. Res.* 15(1):(2014), 1929–1958. DOI: 10.5555/2627435.2670313.

[Sun96] BM Sundheim. Overview of Results of the MUC-6 Evaluation. In: *Proceedings of a Workshop on Held at Vienna, Virginia: May 6-8, 1996*. TIPSTER '96. Vienna, Virginia: Association for Computational Linguistics, 1996, pp. 423–442. DOI: 10.3115/1119018.1119073.

[Uzu+11] Ö Uzuner et al. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. In: *Journal of the American Medical Informatics Association* 18(5):(June 2011), 552–556. ISSN: 1067-5027. DOI: 10.1136/amiajnl-2011-000203.

[Vas+17] A Vaswani et al. Attention is All You Need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[Wal+06] C Walker et al. ACE 2005 Multilingual Training Corpus. https://web.archive.org/web/20220711151407/https://catalog.ldc.upenn.edu/LDC2006T06. [Online; accessed 11-July-2022]. 2006. DOI: 10.35111/xmhb-2b84.

[Wan+22] Q Wang et al. A Comprehensive Survey of Loss Functions in Machine Learning. In: *Annals of Data Science* 9(2):(2022), 187–212. DOI: 10.1007/s40745-020-00253-5.

[WY15] WY Wang and D Yang. That's So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using #petpeeve Tweets. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 2557–2563. DOI: 10.18653/v1/D15-1306.

[WK92] JJ Webster and C Kit. Tokenization as the Initial Phase in NLP. In: *Proceedings of the 14th Conference on Computational Linguistics - Volume 4*. COLING '92. Nantes, France: Association for Computational Linguistics, 1992, pp. 1106–1110. DOI: 10.3115/992424.992434.

[WZ19] J Wei and K Zou. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6382–6388. DOI: 10.18653/v1/D19-1670.

[Wei+13] R Weischedel et al. OntoNotes Release 5.0. https://web.archive.org/web/20220711150624/https://catalog.ldc.upenn.edu/LDC2013T19. [Online; accessed 11-July-2022]. 2013. DOI: 10.35111/xmhb-2b84.

[Wer90] P Werbos. Backpropagation through time: what it does and how to do it. In: *Proceedings of the IEEE* 78(10):(1990), 1550–1560. DOI: 10.1109/5.58337.

[Wol+20] T Wolf et al. Transformers: State-of-the-Art Natural Language Processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6.

[Wu+19] X Wu et al. Conditional BERT Contextual Augmentation. In: *Computational Science – ICCS 2019*. Ed. by JMF Rodrigues et al. Cham: Springer International Publishing, 2019, pp. 84–95. ISBN: 978-3-030-22747-0.

[Xie+20] Q Xie et al. Unsupervised Data Augmentation for Consistency Training. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by H Larochelle et al. 2020.

[YL21] U Yaseen and S Langer. Data Augmentation for Low-Resource Named Entity Recognition Using Backtranslation. In: *CoRR* abs/2108.11703:(2021).

[Yos+12] MA Yosef et al. HYENA: Hierarchical Type Classification for Entity Names. In: *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*. Ed. by M Kay and C Boitet. Indian

Institute of Technology Bombay, 2012, pp. 1361–1370.

[Yu+18] AW Yu et al. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[Zha+20] J Zhang et al. Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[ZYZ20] R Zhang, Y Yu, and C Zhang. SeqMix: Augmenting Active Sequence Labeling via Sequence Mixup. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by B Webber et al. Association for Computational Linguistics, 2020, pp. 8566–8579. DOI: 10.18653/v1/2020.emnlp-main.691.

[ZZL15] X Zhang, JJ Zhao, and Y LeCun. Character-level Convolutional Networks for Text Classification. In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by C Cortes et al. 2015, pp. 649–657.

[Zho+21] R Zhou et al. MELM: Data Augmentation with Masked Entity Language Modeling for Cross-lingual NER. In: *CoRR* abs/2108.13655:(2021).

[Zöl+21] J Zöllner et al. Optimizing Small BERTs Trained for German NER. In: *Inf.* 12(11):(2021), 443. DOI: 10.3390/info12110443.

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Seitens des Verfassers bestehen keine Einwände die vorliegende Bachelorarbeit für die öffentliche Benutzung im Universitätsarchiv zur Verfügung zu stellen.

Jena, den 22.07.202

Robin Erd