h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbmn
FACHBEREICH MATHEMATIK
UND NATURWISSENSCHAFTEN

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Darmstadt
University of Applied Sciences

– Department of
Natural Science and Mathematics
& Computer Science –

Named Entity Recognition on German Medical Data

Thesis for the Award of the Academic Degree
Master of Science (M. Sc.)
in the Course of Studies Data Science

in Cooperation with
Charité – Universitätsmedizin Berlin
and SVA System Vertrieb Alexander GmbH

by

Johanna E. Bohn

Supervisors	:	Prof. Dr. Bettina Harriehausen-Mühlbauer and Prof. Dr. Jutta Groos
Company Supervisors	:	Benjamin Engel and Linda Rebstadt

Date of Issue	:	01. Juni 2022
Date of Delivery	:	16. Nov 2022

DECLARATION

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, 16. Nov 2022

J. Boh

Johanna E. Bohn

ABSTRACT

Medical data can be used to perform analyses and research in the medical domain. However, large percentages of the data remain unstructured and thus unused after they are created. Medical reports are one example of this: they exist to transmit humanly comprehensible information from one doctor to another in natural language. But in order to allow automated systems to take advantage of the information enclosed in natural language, the information must first be extracted.

The task of named entity recognition (NER) is a computational linguistics subtask of information extraction and deals with classifying every word in a document as belonging to one of a set of predefined categories. The categories we extract are diagnosis, treatment, and medication. Our goal is to advance research in the field of German natural language processing (NLP) in the medical domain by finding the best approach to achieve good performance on a benchmark dataset.

We do this by first obtaining and analysing the dataset called Berlin-Tübingen-Oncology corpus (BRONCO). We train, validate, and evaluate six transformer models on it: the monolingual models GBERT, MedBERT, and GELECTRA, as well as the multilingual models mBERT, XLM-RobERTa, and XLM-RoBERTa GER. We train them using both a feature extraction and a fine-tuning approach, and perform hyperparameter optimisation (HPO). We observe that fine-tuning with HPO leads to the best results.

Additionally, we examine the feasibility of performing domain-adaptive pre-training (DAPT) on one model with ambiguous results. Finally, we conduct a detailed inspection of the two best models, i.e. the best monolingual model GELECTRA and the best multilingual model XLM-RoBERTa.

To our knowledge, our fine-tuned GELECTRA model achieves the highest F1-score on the held back BRONCO50 testing dataset with an overall F1-score of 82.2. With that, we manage to outperform the results originally published for the dataset by Kittner et al. by 2.3-7.7 depending on the predefined category [54].

The results show that adapting state-of-the-art models for German medical NER is an open research question with promising results.

ZUSAMMENFASSUNG

Medizinische Daten können für Analysen und Forschung im medizinischen Bereich verwendet werden, jedoch bleibt ein großer Teil der Daten unstrukturiert und somit ungenutzt. Hierfür sind medizinische Berichte wie Arztbriefe ein Beispiel: Sie dienen der Übermittlung von medizinischen Informationen, die in natürlicher Sprache für ärztliches Fachpersonal formuliert wurden. Doch damit die in der natürlichen Sprache enthaltenen Informationen auch von automatisierten Systemen genutzt werden können, müssen diese Informationen zunächst extrahiert werden.

Die Eigennamenerkennung ist ein Teilgebiet der Informationsextraktion im Bereich der Computerlinguistik und befasst sich damit, jedes Wort eines Dokumentes einer zuvor definierten Kategorie zuzuordnen. Die Kategorien, die innerhalb dieser Arbeit zugewiesen werden, sind Diagnose, Behandlung und Medikation. Unser Ziel ist es, die Forschung auf dem Gebiet der natürlichen Sprachverarbeitung in deutscher Sprache im medizinischen Bereich voranzutreiben. Dafür untersuchen wir geeignete Ansätze, um die Vorhersagequalität auf einem Vergleichsdatensatz zu verbessern.

Hierzu analysieren wir zunächst den Datensatz namens Berlin-Tübingen-Oncology Corpus (BRONCO). Mit Hilfe dessen trainieren, validieren und evaluieren wir sechs Transformer-Modelle: die deutschsprachigen Modelle GBERT, MedBERT und GELECTRA sowie die mehrsprachigen Modelle mBERT, XLM-RoBERTa und XLM-RoBERTa GER. Wir trainieren die Modelle sowohl mit einem Merkmalsextraktions- als auch mit einem Feinabstimmungsansatz und führen eine Hyperparameter-Optimierung durch. Die Ergebnisse zeigen auf, dass der Feinabstimmungsansatz mit Hyperparameter-Optimierung zu den besten Ergebnissen führt.

Außerdem untersuchen wir die Umsetzbarkeit von domänenadaptivem Vortrainieren auf einem der Modelle. Aufgrund der geringen Menge an Daten zum Vortrainieren konnten wir bei diesem Ansatz keine signifikanten Verbesserungen feststellen. Schließlich führen wir eine detaillierte Untersuchung der Ergebnisse des besten deutschsprachigen Modells GELECTRA und des besten mehrsprachigen Modells XLM-RoBERTa durch.

Unseres Wissens nach erzielt GELECTRA den bisher höchsten F1-Wert auf dem zurückgehaltenen BRONCO50-Testdatensatz mit einer Gesamtpunktzahl von 82.2 Prozent. Wir übertreffen damit die Ergebnisse, die von Kittner et al. für diesen Datensatz veröffentlicht wurden, je nach Kategorie um 2.3-7.7 Prozentpunkte [54].

Die Ergebnisse zeigen, dass die Adaption von Modellen auf dem aktuellen Stand der Forschung an die deutsche medizinische Eigennamenerkennung ein offenes Forschungsfeld mit vielversprechender Zukunft ist.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

GDPR	general data protection regulation
NER	named entity recognition
NE	named entity
POS	part of speech
STTS	Stuttgart-Tübingen-TagSet
NLP	natural language processing
TP	true positive
FP	false positive
TN	true negative
FN	false negative
LSTM	long short-term memory
BERT	bidirectional encoder representations from transformers
mBERT	multilingual BERT
ELECTRA	efficiently learning an encoder that classifies token replacements accurately
RoBERTa	robustly optimised BERT approach
NSP	next sentence prediction
MLM	masked language model
CRF	conditional random field
CNN	convolutional neural network
BRONCO	Berlin-Tübingen-Oncology corpus
WE	word embedding
HPO	hyperparameter optimisation
PBT	population based training
TPE	tree-structured parzen estimator
BO	bayesian optimisation
DAPT	domain-adaptive pre-training
NN	noun
NE	proper noun, name
ART	article
ADJA	attribute adjective
APPR	preposition

VVFIN	finite modal verb
XY	non-word

Part I

INTRODUCTION

1 MOTIVATION AND PROBLEM STATEMENT

Medical data enables the healthcare system and research to achieve medical progress, better prevention, and better treatment of patients. In order to unlock the data's potential, prerequisites to make it available and accessible must be established [5].

For years, digitisation in the field of medicine and healthcare has been a widely discussed topic in Germany [29]. In 2015, the German government passed a law regarding secure digital communication and applications in the field of healthcare [12]. It encouraged, among several other strategies to advance the creation of a secure telematics infrastructure, the introduction and use of medical applications like the electronic medical report *eArztbrief* and the electronic patient chart *ePA* [12].

While these advancements are promising, a big factor when working with German medical data are the restrictions in place to protect it: In addition to the general data protection regulation (GDPR) of 2018 that solidifies the protection of natural persons with regard to the processing of personal data, a law on the protection of patient data became effective in 2021 [13].

Even when data is available in a digital form and authorised to be utilised for research, the information it contains may not be accessible for computer systems: 80% of medical data remains unstructured and thus unused after it is created, e.g. text, image, and signal [55]. A lot of information is enclosed in medical texts like electronic medical reports [33] which computational systems cannot easily work with. The main purpose of the report is to transmit humanly comprehensible diagnostic and treatment information from one doctor to another. As such, it is usually the most comprehensive and detailed depiction of a patients course of treatment [90]. However, as it is a form of communication between people, it cannot be directly utilised by computer systems for research. Instead, the unstructured data needs to be converted into a structured format to be taken advantage of by automated systems. Only then can automatic analyses be performed.

To make all that data usable, the most important information in medical reports needs to be extracted [15]. An example is illustrated in Figure 1.1, where the most important information consists of the diagnosis of an illness by a doctor (DIAG), its treatment (TREAT), and the medication the patient was administered (MED).

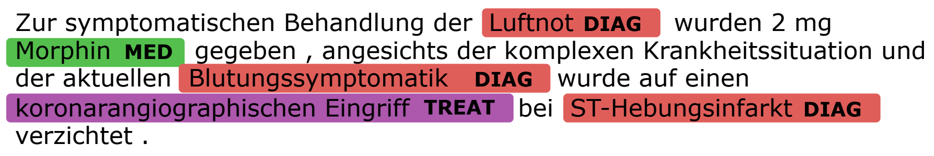

Zur symptomatischen Behandlung der Luftnot **DIAG** wurden 2 mg Morphin **MED** gegeben , angesichts der komplexen Krankheitssituation und der aktuellen Blutungssymptomatik **DIAG** wurde auf einen koronarangiographischen Eingriff **TREAT** bei ST-Hebungsinfarkt **DIAG** verzichtet .

Figure 1.1: NER example taken from the BRONCO dataset [54].

Based on those extracted entities, medical research is able to for instance analyse the correlation between specific medications and symptoms and gain knowledge beyond limited pharmaceutical studies [16].

The process of classifying every word in a document as belonging to one of a set of predefined categories is called NER and a computational linguistics subtask of information extraction [9].

In this thesis, we will perform NER on a set of German medical reports to train a model that extracts the entities *diagnosis*, *treatment*, and *medication*. We will compare different model types and training regimes, and evaluate and discuss the results. Our goal is to advance research in the field of German NLP in the medical domain by finding the best approach to achieving good performances on a benchmark dataset.

2 STRATEGY

In order to attain the goal of finding the best approach to achieving good performances on a benchmark dataset, a set of steps needs to be accomplished:

1. Obtaining the dataset and preprocessing the data
2. Choosing, training, and optimising models
3. Evaluating and discussing results

2.1 OBTAINING THE DATASET AND PREPROCESSING THE DATA

As already mentioned in Section 1, obtaining a dataset for a certain task can be difficult. Once it has been acquired, the data needs to be preprocessed for the task at hand. That might include examining the data for suitability, performing exploratory data analysis, and formatting it for use in a pipeline. Additionally, the dataset needs to be prepared for training and evaluating models on it by splitting it into training, validation, and test sets (see Section 9).

2.2 CHOOSING, TRAINING, AND OPTIMISING MODELS

Based on the current state of research (see Section 7), there are different avenues to explore. Mainly, transformer models lend themselves to NER as either monolingual or multilingual options. They can be trained in different ways, i.e. feature extraction and fine-tuning, which are both suitable for different situations. In addition to training the models, hyperparameter optimisation (HPO) needs to be performed in order to optimise the hyperparameters to gain more optimised models.

2.3 EVALUATING AND DISCUSSING RESULTS

In order to evaluate their performance, the trained models also need to be studied with respect to one or multiple metrics and under several aspects. Those can include the kind of training performed, if HPO improved the results, if monolingual models perform better than multilingual models, etc. These factors need to be discussed and brought to a final conclusion about how a good result for the task at hand can be achieved.

3 STRUCTURE

This thesis is organised in five parts as follows. Part i introduces the problem, the motivation behind solving it, and the strategy with which we aim to solve it.

Then, Part ii gives an overview of the fundamentals of NLP and the methods in use for NLP in general, and specifically of NER. The focus here will be deep learning. Based on that knowledge, it will introduce the context of the current state of research and related work for the task at hand.

The description of the implemented system follows in Part iii by starting with the utilised dataset and the hardware and software leveraged for computation of the chosen models. Those models are introduced afterwards and the avenue of hyperparameter optimisation (HPO) is explained. This part describes the theoretical approach of how we are trying to solve the task at hand by formulating the experiments we perform.

Part iv evaluates the results of conducting the experiments in order to compare and discuss them.

Finally, Part v completes the thesis with a summary of what we learned, the conclusion this brings us to, and an outlook for future work that could be done.

Part II

FUNDAMENTALS

4 NATURAL LANGUAGE PROCESSING

In this part, we first focus on the theoretical background that is necessary to approach an NLP task by employing modern methods. We start by defining key concepts of NLP and deep learning and explaining why we need them. Then, we give context in the form of an overview over the related work of NER and which methods are currently in use. In doing so, we can focus on how we utilise those methods when we discuss the details of the developed system in Part iii.

Natural language processing (NLP) is a subdiscipline of computer science providing a bridge between natural languages and computers. Kamath et al. define it as follows [51]:

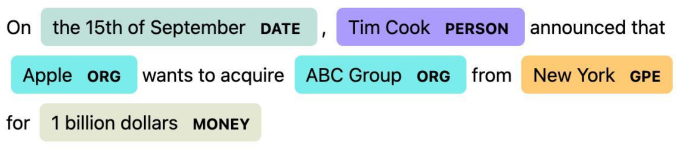
Natural language processing (NLP) seeks to map language to representations that capture morphological, lexical, syntactic, semantic, or discourse characteristics that can then be processed by machine learning methods.

It helps to empower machines to understand, process, and analyse human language [64]. Applications range from machine translation over semantic parsing to information extraction. The latter identifies structured information in *unstructured* data. Its subtasks include named entity recognition (NER), relation extraction, coreference resolution, and more.

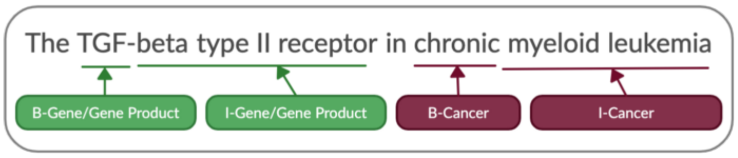
NER aims to locate and categorise NEs into predefined categories [89]. A NE was defined as "[...] a proper noun, serving as a name for something or someone" by Petasis et al. [73]. Focusing on the term *named*, it was further restricted to "only those entities for which one or many *rigid designators* stands for the referent" [69]. Those rigid designators include "proper names as well as certain natural kind terms like biological species and substances" [69]. However, those definitions can be loosened for practical reasons like including temporal or numerical expressions [69]. Although there are multiple definitions of NEs and NER, a working definition can be delineated as follows [60]:

A **named entity (NE)** is a word or a phrase that clearly identifies one item from a set of other items that have similar attributes.
Named entity recognition (NER) is the process of locating and classifying NEs in text into predefined entity categories.

There are two categories of NEs: generic, e.g. person and location, and domain-specific, e.g. proteins, enzymes, and genes (see Figure 4.1) [60]. The NEs examined in this thesis belong to the second group as they are part of the medical domain.


On the 15th of September DATE , Tim Cook PERSON announced that
Apple ORG wants to acquire ABC Group ORG from New York GPE
for 1 billion dollars MONEY

(a) Generic



(b) Domain-Specific

Figure 4.1: Generic [11] versus domain-specific NEs [72].

Regardless of the kind of NE explored, there exist different streams of techniques applied in NER [60]: Rule-based approaches, deep-learning based approaches, and many more.

One of the most appealing approaches in the NLP domain is deep learning. It has already demonstrated superior performance in adjoining fields like Computer Vision and Speech Recognition [89]. In addition, multiple surveys have identified several deep learning approaches as the current state of the art for different subtasks of NLP including NER [60, 89, 96]. Thus, we are focusing on that technique in this thesis.

4.1 PREPROCESSING

In order to apply deep learning techniques to a text, it usually needs to be preprocessed [65, p. 10]. Most of the data preprocessing steps are specified by the model to be used (see Section 6.3), which means there is a limit on how much we can influence those steps.

We describe tokenisation in more detail by way of example for some challenges posed by working with natural language.

Tokenisation is the task of chopping up a character sequence into pieces called *tokens* while sometimes dropping certain characters like punctuation (see Figure 4.2) [63].

Input: Friends, Romans, Countrymen, lend me your ears;

Output: $\boxed{\text{Friends}}$ $\boxed{\text{Romans}}$ $\boxed{\text{Countrymen}}$ $\boxed{\text{lend}}$ $\boxed{\text{me}}$ $\boxed{\text{your}}$ $\boxed{\text{ears}}$

Figure 4.2: Example of tokenisation [63].

Tokens may be words, characters, or even bytes, but must always be discrete entities [37, p. 461]. A common strategy is to separate tokens by whitespace to receive words. However, even when using tokens that correspond to words, there are challenges in how characters are separated (see Figure 4.3).

Input: aren't

Output Options: $\boxed{\text{aren't}}$, $\boxed{\text{arent}}$, $\boxed{\text{are}}\ \boxed{\text{n't}}$, $\boxed{\text{aren}}\ \boxed{\text{t}}$

Figure 4.3: Challenge of tokenisation [63].

Because of those ambiguities, various methods to achieve appropriate tokens have been proposed [40]:

1. rule-based tokenisation
2. dictionary-based tokenisation
3. supervised tokenisation with neural networks
4. unsupervised tokenisation

Which kind of tokenisation strategy is chosen depends not only on the downstream task it is needed for, but also on the downstream model [40]. Additionally, the issues of tokenisation can be language-specific [63].

5 DEEP LEARNING

Now that we have covered NLP basics, we can go on to the background of the aforementioned deep learning approach. We aim to first give an understanding of deep learning in general based on Kinsley et al. [53] before we bring techniques used for NLP into focus.

Torfi et al. describe deep learning as follows [89]:

> **Deep learning** refers to applying deep neural networks to massive amounts of data to detect and analyse important structures/features in order to learn a procedure aimed at handling a task.

Deep neural networks are a subset of neural networks, from which they are delimited by having two or more *hidden layers* where shallow neural networks only have one. As such, they are a part of machine learning and hence a branch of artificial intelligence (see Figure 5.1) [53, p. 8].

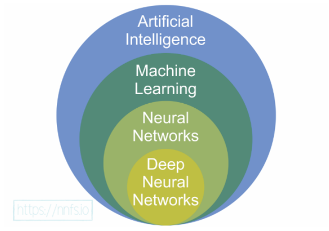


Figure 5.1: Breakdown of the hierarchy from artificial intelligence to deep neural networks [53, p. 8].

Just like neural networks, deep neural networks are "inspired by the organic brain, translated to the computer" [53, p. 10] as can be seen in Figure 5.2.

But only when hundreds, thousands, or more neurons are connected, can they produce results that frequently outperform any other machine learning method [53, p. 10]. Figure 5.3 depicts a basic example of a neural network composed of artificial neurons with input layer, two hidden layers, and output layer.

The input layer receives the actual input data, preprocessed and in a numeric form. The hidden layers perform the feature detection whose result is then processed and returned by the output layer [37, 53, pp. 16, 6]. Since we are looking at a token classification task, we will only be considering common classification techniques. For those, the output layer often has as many neurons as the training dataset has classes [53, p. 16].

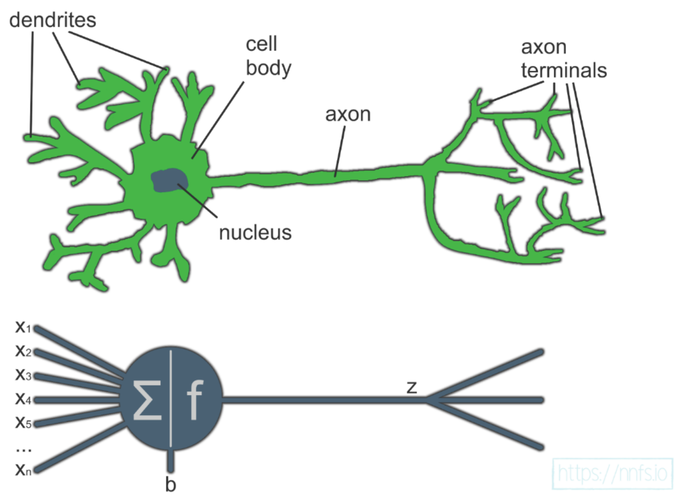


Figure 5.2: Comparison of a biological neuron and an artificial neuron [53, p. 10]. The dendrites act as input layer, cell body and nucleus as activation function, and the axon terminals as output layer of the (artificial) neuron.

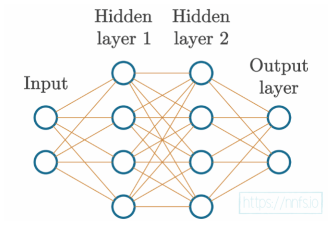
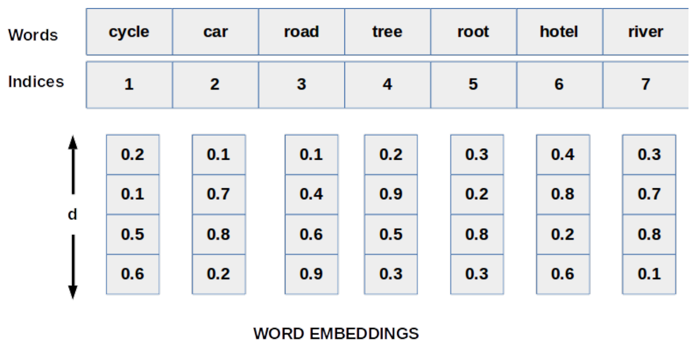


Figure 5.3: Example of a basic neural network [53, p. 16].

5.1 EMBEDDING

In order to get tokens created by a tokeniser into a form suitable for a neural network, they need to be assigned numerical values [37, p. 212]. This is done by referring to each token by the index this token has in a vocabulary file. For each of those indices, the *embedding* is a vector in a *d*-dimensional space [75]. Figure 5.4 shows an example of what a word embedding might look like.



Words	cycle	car	road	tree	root	hotel	river
Indices	1	2	3	4	5	6	7
d	0.2	0.1	0.1	0.2	0.3	0.4	0.3
	0.1	0.7	0.4	0.9	0.2	0.8	0.7
	0.5	0.8	0.6	0.5	0.8	0.2	0.8
	0.6	0.2	0.9	0.3	0.3	0.6	0.1

WORD EMBEDDINGS

Figure 5.4: Example of a possible word embedding [75].

Token embeddings are series of real numbers that represent tokens. As such, similarity of tokens can be computed by calculating the distance of their vectors [91].

5.2 TRAINING

This section introduces the concept of how a deep neural network, i.e. a model, learns. As mentioned before, the goal of deep learning is to formulate a solution to a problem by detecting and analysing important structures or features in the data [89]. This is done by a series of hidden layers that extract increasingly abstract features from the input data. Figure 5.5 illustrates an example of a computer vision classification task. Analogous for NLP, morphological information is captured by the word embedding layer, local syntax by lower layers, and longer range semantics like coreference by the upper layers [78, p. 75].

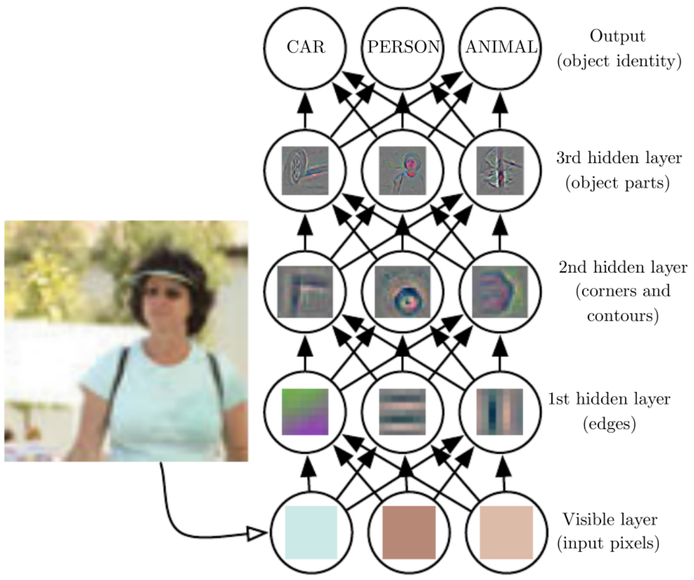

CAR
PERSON
ANIMAL
Output (object identity)
3rd hidden layer (object parts)
2nd hidden layer (corners and contours)
1st hidden layer (edges)
Visible layer (input pixels)

Figure 5.5: Example of visual features extracted for a computer vision classification task [37, p. 6].

Based on the observed data, the model aims to determine which underlying concepts are useful for explaining the relationships present in that data [37, p. 6]. It then adjusts its *parameters* or *weights*, which are values that control the behaviour of the system. They determine how each feature in the data affects the prediction [37, p. 107].

The data that is utilised for the extraction of features is called the **training dataset**. It consists of many **training samples** [68, p. 3].

How the feat of training is accomplished in detail lies outside the scope of this thesis. In it's core, a classification task's goal like NER is to optimise, i.e. minimise, the difference between a given and a predicted label by adapting its model's parameters [85]. This training error gives an idea of how well the model is fit to the training data. It is related to the generalisation error which is a measure of how well the model can generalise from seen to unseen data. How a model's generalisation error typically evolves during training can be seen in Figure 5.6. It illustrates what happens to training and generalisation error when we increase the capacity of a model. A good generalisation error is achieved, when the gap between training and generalisation error is small. In the underfitting regime, the gap is small, but both training error and generalisation error are high. When we increase capacity, the gap between the errors increases. The optimal capacity is reached at a point where that gap is compensated by the decrease of the training error. Past that, the capacity of the model is too large and the size of the gap outweighs the decrease in training error. This is the overfitting regime. A detailed explanation can be found in Kinsley et al. and Goodfellow et al. [37, 53].

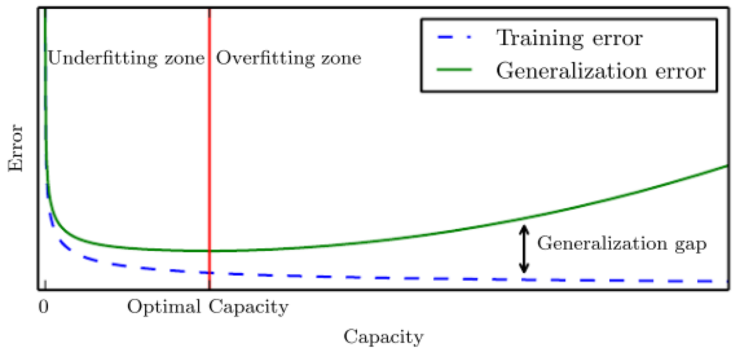


Figure 5.6: Typical relationship between capacity and error from underfitting to overfitting regime [37, p. 115].

There are, however, different ways in which a model can be trained for a target task like NER. In the following sections, we will discern the differences between several techniques in regard to training a deep learning model:

- supervised, unsupervised, and semi-supervised learning
- traditional machine learning and transfer learning

5.2.1 *Supervision*

Supervision can be divided into several types, with each being suitable for a different machine learning scenario [68, p. 7]:

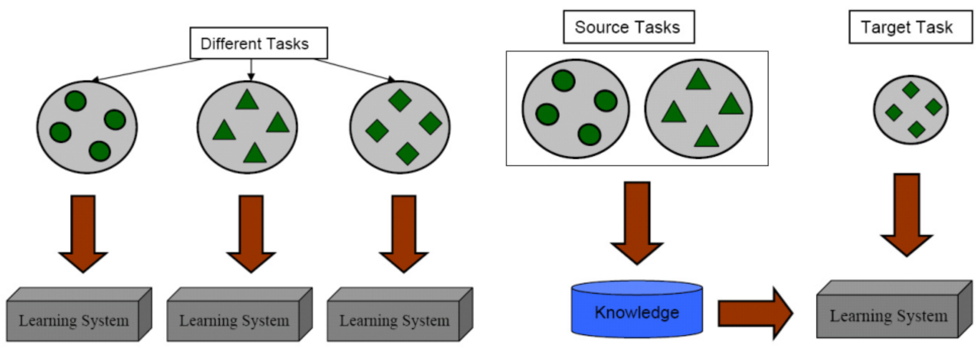
- Supervised learning: labelled data points are available as training data e.g. for classification, regression, and ranking problems.
- Unsupervised learning: only unlabelled training data exists
  e.g. for clustering and dimensionality reduction.
- Semi-supervised learning: a set of training samples consisting of both labelled and unlabelled data is provided
  e.g. for classification, regression, and ranking tasks.

Predictions are then made for all unseen points.

As we can see, the kind of supervision is reliant on the available data. Subsequently, the kind of supervision limits the applicability of algorithms for the task at hand. Based on the dataset and techniques we utilise (see Sections 6.2 and 9), we focus on supervised learning from this point forward in this thesis.

5.3 TRANSFER LEARNING

Where traditional machine learning techniques try to learn each task from scratch, transfer learning techniques try to transfer knowledge from previously learned source tasks to a target task (see Figure 5.7). In doing so, a model does not need to learn everything from the data of the target task, reducing the amount needed for a good performance. Thus a significant amount of labelling effort for the target task can be saved, which is particularly advantageous in cases where annotated data is hard to come by [71].



(a) Traditional Machine Learning (b) Transfer Learning

Figure 5.7: Comparison between traditional machine learning and transfer learning [71].

Ruder et al. define transfer learning the following way [79]:

"**Transfer learning** is a means to extract knowledge from a source setting and apply it to a different target setting." [79]

There are many scenarios that ask for different kinds of transfer learning. Arguably, the one that is used most frequently in NLP and machine learning is the branch of *sequential transfer learning* [78, p. 63]. Here, the source and target tasks are different and one is performed after the other instead of at once like for other branches, e.g. multi-task learning. The goal is to "transfer information from the model trained on the source task to improve performance of the target model" [78, p. 63]. Typically, sequential transfer learning consists of two stages [78, p. 64]:

1. *Pre-training* phase: a model is trained on the source task.

2. *Adaptation* phase: knowledge of the trained model is transferred to the target task.

Generally, training the source model is expensive and only needs to be done once, while adaptation to the target task is fast [78, pp. 63–64]. For that reason, it is common for the first stage to choose a source task which will enable learning a representation of the data that will be useful for a wide range of target tasks. A *representation* in this context means a set of compressed latent features of the input data [86] [78, p. 34].

Within the space of NLP, the expectation is that leveraging the underlying structure of language will improve generalisation to those target tasks [78, p. 64]. A pre-trained model in the context of modern NLP is called a *pre-trained language model* [78, p. 75].

For the second stage, there exist two main ways to adapt a pre-trained model to a target task [78, p. 77]:

- feature extraction: all parameters in the network are frozen and the pre-trained representations are used in downstream models
- fine-tuning: the parameters in the network are used as initialisation for the model for the downstream task

Both generally achieve a similar performance, but because parameters are adjusted in the fine-tuning approach, it needs more resources. It is said that feature extraction performs better when source and target tasks are distant, while fine-tuning performs better when they are similar [78, p. 77].

Pre-Training can either be performed in one language or multiple different languages. Consequently, the adaptation phase should only be implemented for those languages that the model was pre-trained on. The resulting language models are either monolingual [6, 14, 28] or multilingual models [62].

5.3.1 *Loss*

Since classification is an optimisation task, we need a function to optimise, i.e. minimise, the error/loss [85]. This is called the *loss function* [37, p. 82]. Mohri et al. define it as follows [68, p. 4]:

> The **loss function** "measures the difference, or loss, between a predicted label and a true label" [68, p. 4].

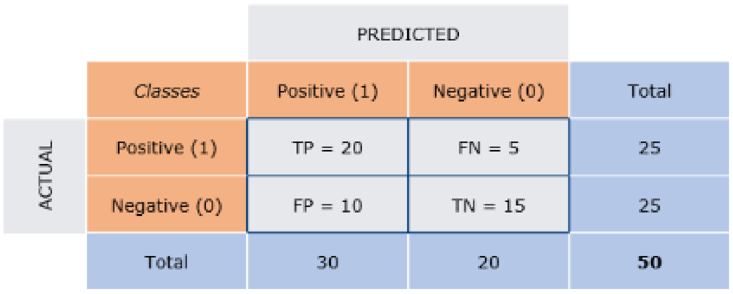
5.4 metric

Apart from the loss function that can act as a measure for the error during training, there are additional performance indicators called *metrics*. They do not influence the optimisation process, but are useful to evaluate and compare different classification models or machine learning techniques [38].

Based on Grandini et al., we formulate classification as a prediction task where available data X is used to obtain the best prediction $\hat{Y}$ of the outcome variable Y [38].

5.4.1 *Binary Case*

In order to define the more advanced metrics, we start by introducing *binary precision* and *binary recall* with the aid of a confusion matrix (see Figure 5.8) [38]. The matrix shows a binary classification task, i.e. with two classes *positive* and *negative*, with 50 samples. Out of those, 35 have been correctly predicted, i.e. $\hat{Y} = Y$: 20 as positive (true positive (TP)) and 15 as negative (true negative (TN)). 15 have been incorrectly predicted, i.e. $\hat{Y} \neq Y$: 10 as positive when they were negative (false positive (FP)), and 5 as negative when they were positive (false negative (FN)).



		PREDICTED		
	Classes	Positive (1)	Negative (0)	Total
ACTUAL	Positive (1)	TP = 20	FN = 5	25
	Negative (0)	FP = 10	TN = 15	25
	Total	30	20	**50**

Figure 5.8: Example of a two-class confusion matrix with true positives (TPs), false positives (FPs), false negatives (FNs), true negatives (TNs) [38].

(Binary) Precision is TPs divided by the total amount of samples that were predicted as positive. It is a value denoting the trust we can have in the model when it predicts a sample as positive [38]:

$$\text{Precision}_{\text{binary}} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.1}$$

(Binary) Recall is TPs divided by the total amount of samples that are actually positive. It denotes the ability of the model to find all positive samples in the dataset [38]:

$$\text{Recall}_{\text{binary}} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.2}$$

In other words, precision refers to the percentage of a system's results that are correctly recognised. Recall refers to the percentage of total entities correctly recognised by a system [60]. Both measures take on values in the range $[0;1]$ [38].

A common metric used for NER is the *(binary) F1-Score* [60]. It aggregates precision and recall under the concept of harmonic mean [38]:

$$\text{F1}_{\text{binary}} = \frac{2}{\text{precision}^{(-1)} + \text{recall}^{(-1)}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5.3}$$

The score can be understood as the weighted average between precision and recall, aiming to find the best trade-off between the two [38]. As such, it rewards models that have similar precision and recall values. It is delimited by 0 and 1, meaning that its best value is at 1 and its worst at 0. However, the harmonic mean "tends to give more weight to lower values", which makes the score decrease immensely when either precision or recall assume a value close to 0 [38].

Following the confusion matrix in Figure 5.8, precision, recall and F1-Score result in [38]:

$$\begin{aligned} \text{Precision}_{\text{binary}} &= \frac{20}{30} \approx 0.66 \\ \text{Recall}_{\text{binary}} &= \frac{20}{25} = 0.8 \\ \text{F1}_{\text{binary}} &= 2 \cdot \frac{0.66 \cdot 0.8}{0.66 + 0.8} \approx 0.72 \end{aligned} \tag{5.4}$$

5.4.2 *Multi-Class Case*

In order to evaluate a model classifying more than two classes, we need to use a multi-class version of the F1-Score: either the *micro-averaged F1-Score* or the *macro-averaged F1-Score*. In turn, we require the accompanying measures of precision and recall for K classes:

$$\begin{aligned} \text{averaged Precision}_{\text{macro}} &= \frac{\sum_{k=1}^{K} \text{Precision}_k}{K} \\ \text{averaged Recall}_{\text{macro}} &= \frac{\sum_{k=1}^{K} \text{Recall}_k}{K} \\ \text{F1}_{\text{macro}} &= 2 \cdot \frac{\text{averaged Precision}_{\text{macro}} \cdot \text{averaged Recall}_{\text{macro}}}{\text{averaged Precision}_{\text{macro}} + \text{averaged Recall}_{\text{macro}}} \end{aligned} \tag{5.5}$$

In macro-averaging, the performance is collected for each class and then averaged over classes. In micro-averaging, all decisions are collected and precision and recall computed from that complete collection [50, p. 67].

When we remember the formulas for precision 5.1 and recall 5.2, we see that they only differ in their denominator. However, in a multi-class setting, if there is a FP, there is also always a FN and vice versa [39], e.g.:

- class A is predicted and true label is B: FP for A and FN for B
- class A is predicted and true label is A: TP for A and TN for B

In conclusion, there is no possibility that would increase only FP or FN but not both [39].

Because we are considering the whole dataset and not just a class of elements, the micro-averaged precision and recall give us:

$$\text{averaged Precision}_{\text{micro}} = \text{averaged Recall}_{\text{micro}} = \frac{\sum_{k=1}^{K} \text{TP}_k}{|\text{elements}|} \tag{5.6}$$

And since the harmonic mean of two equal values is the value itself, the result is [38]:

$$\text{F1}_{\text{micro}} = \frac{\sum_{k=1}^{K} \text{TP}_k}{|\text{elements}|} \tag{5.7}$$

In conclusion, a macro-average is an average measure of the average precision and recall of the classes, with each class having the same weight. Contrarily, a micro-average gives each sample the same importance, and hence different weights to different classes based on their frequency in the dataset [38].

We scale the F1-score by a factor of 100 for readability in the text of this thesis, transforming its range from 0.0-1.0 to 0.0-100.0.

5.5 VALIDATION AND TESTING

We have already established that a model needs data to train its features, i.e. the model's parameters. However, we also want to asses the improvement of the model during training with metrics and maximise that improvement by tweaking the model's hyperparameters. Figure 5.9 illustrates the distinction between the model's parameters and hyperparameters, i.e. settings to control the behaviour of the model. Finally, we aim to estimate the generalisation error [37, pp. 120–121]. This Section introduces the means with which those goals can be achieved.

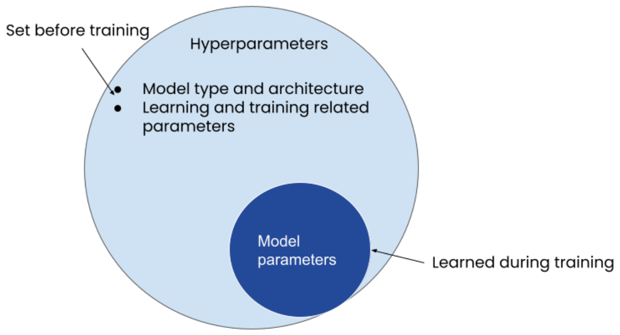


Figure 5.9: Distinction between model parameters and hyperparameters [88].

5.5.1 *Datasets*

The techniques to attain those goals are based around separating the data into three different datasets (see Figure 5.10).

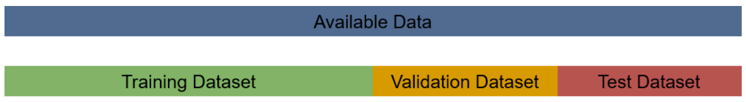


Figure 5.10: Separation of the available data into training, validation, and test subset in an exemplary 50:25:25 split.

We have already defined the *training dataset* in Section 5.2:

> The data that is utilised for the extraction of features is called the **training dataset**. It consists of many **training samples** [68, p. 3].

For the first of the two challenges, i.e. measuring the training improvement under different combinations of hyperparameters and evaluating those combinations, we can utilise a *validation dataset* of samples that the training algorithm does not observe [37, p. 121]:

> The **validation dataset** allows the tuning of hyperparameters of a learning algorithm. The **validation samples** are used to select appropriate values for the model's hyperparameters [68, pp. 3–4].

This can happen during or after the training process [37, p. 121].

For the second challenge, i.e. measuring the generalisation error, we want to determine how well the model will work in the real world by confronting it with data it has not seen before, called the *test dataset* [37, p. 104].

> The **test dataset** is used to evaluate the performance of a learning algorithm on unseen data. That is achieved by comparing each **test sample's** true label to the label predicted by the algorithm [68, p. 4].

It is important that this happens only after all hyperparameter optimisation is complete so that the test dataset does not influence choices about the model [37, p. 121]. If it did, this would be considered *data leakage*, i.e. utilising information that should not legitimately be available [52].

5.5.2 *Splitting*

In order to achieve accurate estimates on how well the model will perform on new data, i.e. the generalisation error, a few factors need to be addressed [95]: the distribution of the datasets, the method used for splitting, and the balance between training, validation, and test dataset.

Distribution

We usually do not know the true underlying distribution of the data. Because of that, it is impossible to tell how well the estimated predictive performance on the test dataset matches it [95]. The general assumption is that the "measured performance" on the test dataset "is an unbiased, accurate estimator for the model performance on all unknown samples coming from the same distribution of the training/test dataset" [95]. Following that assumption, resampling from the dataset is used to approximate similar distributions for the dataset splits [95].

Method
The basic idea for *cross validation*, a class of model evaluation methods, is to train a model on one part of the data and test the performance on another. The simplest version is the *holdout method*, where the data is separated by holding out a part of the data as validation or test dataset [37, p. 123] [82]. Some more advanced methods for smaller datasets include repeating the training, validating, and testing on different subsets or splits of the original dataset and averaging the results [37, pp. 122–123].

Balance
Regardless of the method utilised for splitting, a good balance between training, validation and test dataset is required to have a stable estimation of the model performance [95]. A commonly used strategy is allocating two thirds of samples for training [30].

Xu et al. claim that there is "no clear evidence suggesting which method/ parameter combination would always give significantly better results than others" [95]. Thus, the method of data splitting and its parameters "cannot be decided a priori and would be data dependent" [95].

6 DEEP LEARNING FOR NLP

With the basics of deep learning out of the way, we can now focus on the techniques used for NLP.

6.1 ENCODER-DECODER

Many applications of NLP rely on sequence-to-sequence techniques with inputs and outputs of variable lengths [98, p. 374]. For these types of inputs and outputs, an architecture with two major components can be designed: the *encoder-decoder* architecture (see Figure 6.1). The *encoder* takes an input sequence of variable length one sample at a time and transforms it into a state with fixed shape, called *compressed latent features* or *representation* [78, p. 34]. The *decoder* maps that encoded state to an output sequence of variable length [86, 98, p. 374].

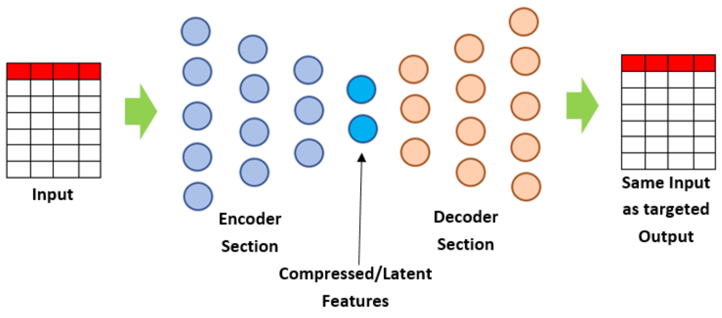


Figure 6.1: The illustrated encoder-decoder architecture [22].

Both of those steps are performed by two separate architectures, namely two LSTMs, in the original paper (see Figure 6.2) [86]. The training objective in that paper encourages an LSTM to "find sentence representations that capture their meaning, as sentences with similar meanings are close to each other while different sentences' meanings will be far" [86].

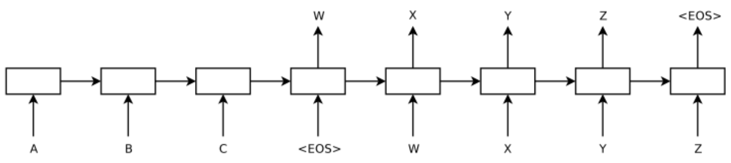


Figure 6.2: A LSTM reads the string "ABC" in reverse, i.e. as "CBA" and another produces "WXYZ" as output sentence, stopping at the end-of-sentence token <EOS> [86].

A challenge in tasks like machine translation is learning dependencies over a long range [92]. Sutskever et al. theorise that the reversion of the input sequences performed in their model makes it easier to "establish communication" between input and output sequence. They reverse the input sentence and concatenate it with the output sentence, e.g. "ABC<EOS>WXYZ<EOS>" turns into "CBA<EOS>WXYZ<EOS>". This way, the "average distance between corresponding words in the source and target language is unchanged" [86], but the distance between the first few words in the input language and the first few words in the output language is reduced. This did not only improve performance on short sentences, but also long sentences [86].

6.2 ATTENTION IN TRANSFORMERS

As a next step, *attention* mechanisms make it possible to model dependencies regardless of their distance in the input or output sequence [92].

While the inherent sequential nature of LSTM architectures makes parallelisation within training samples impossible, Vaswani et al. introduce the *transformer* architecture that relies solely on an attention mechanism instead of LSTMs and thus allows for more parallelisation. This attention mechanism is able to draw global dependencies between input and output, forfeiting the need for close proximity of input and output tokens that are susceptible to each other [92].

The core innovation of transformers compared to sequential architectures lies in the attention mechanism that allows the decoder to look back at the encoder states [77]. Vaswani et al. describe attention mechanisms as follows [92]:

"An **attention** function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key." [92]

Their specific attention mechanism is a *multi-head attention* that performs *scaled dot-product attention* on each of several different linear projections on the queries, keys, and values [92]. In combination, they are used in three different ways:

- The "encoder-decoder" attention layers "allow every position in the decoder to attend over all positions in the input sequence" [92].
- The encoder's self-attention layers make it possible for the encoder to "attend to all positions in the previous layer of the encoder" [92].
- The decoder's self-attention layers "allow each position in the decoder to attend to all positions in the decoder up to and including that position" [92]. Illegal tokens, i.e. tokens that the decoder is not allowed to see yet, are hidden using *masking* tokens.

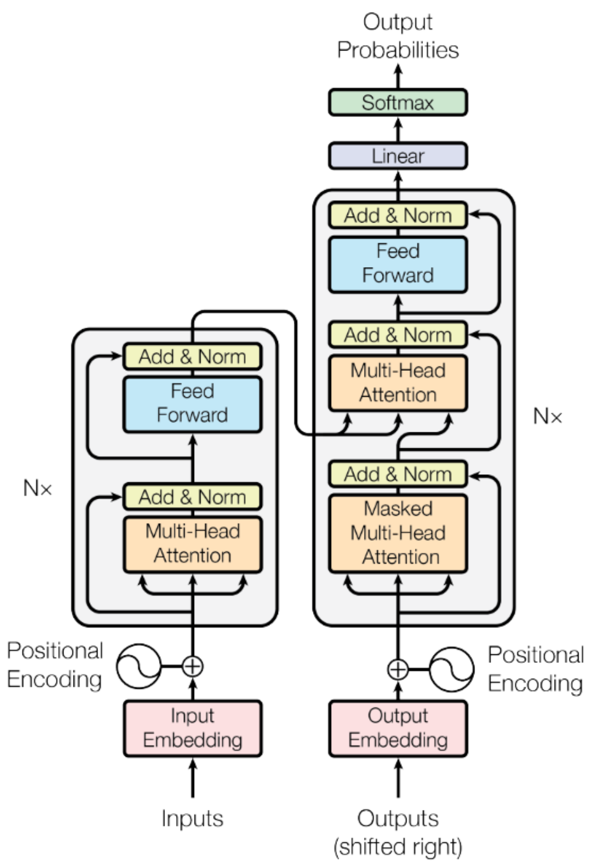


Figure 6.3: Architecture of the transformer model [92].

Figure 6.3 shows the architecture of the transformer model whose description we simplify based on Vaswani et al. for better understanding [92].

The computation graph depicts the encoder and decoder part of the architecture, left and right respectively. Both consist of stacked self-attention and point-wise, fully connected layers.

After tokenisation (see Section 4.1), the input tokens are converted to vectors of the dimension the model requires by learned embeddings. Then, a positional encoding is added to the input embeddings for the model to make use of the order of the sequence. The result is then put into a stack of $N = 6$ identical layers of the encoder. Each consists of a multi-head self-attention mechanism and a fully connected feed-forward network with residual connections and their normalisation layers.

The decoder also consists of a stack of $N = 6$ identical layers almost like the encoder, but with one additional multi-head attention layer that attends over the output of the encoder stack. The multi-head attention that is similar to the encoder's is additionally modified to prevent positions from attending to subsequent, i.e. illegal, positions by masking them. It is called the *masked multi-head attention*.

The vectors are finally run through a linear transformation and softmax function to compute the next-token probabilities for the translation [92].

Sutskever et al. already recognised that the length of paths in the network is a big influence on the model's ability to learn long-range dependencies [86]. The self-attention deployed in Vaswani et al. leads to a maximum path length of constant number between all positions, thus making it easier for the model to learn those long-range dependencies [92]. Another benefit of self-attention is the added interpretability of models: attention heads learn to perform different tasks, with many appearing to "exhibit behaviour related to the syntactic and semantic structure of the sentences" [92].

6.3 PRE-TRAINED LANGUAGE MODEL: BERT

The attention mechanism explained in the last section has furthered research on many challenging NLP tasks. But while the architecture can be applied to many tasks, each one still needs task-specific training with datasets of thousands of samples [10].

Utilising transfer learning techniques (see Section 5.3) and pre-training language models before applying them to downstream tasks can improve many NLP tasks [28]. Depending on the task, it can be enough to only utilise one half of the transformer, i.e. either encoder or decoder, depending on if an understanding of the input or a generation of output sequences is needed [24, 31].

Devlin et al. introduce a language representation model that we examine in this section: bidirectional encoder representations from transformers (BERT) [28]. It is an encoder model, meaning that it only contains the encoder, but not the decoder part of the encoder-decoder architecture [28]. Models like this can encode information from the input into a representation, but cannot generate sequences as outputs [86, p. 374]. This makes them a good fit for tasks that require understanding the input like NER [31, 43]. BERT achieves state-of-the-art results on more than 10 NLP tasks. It accomplishes that feat by utilising a masked language model (MLM) objective in combination with a next sentence prediction (NSP) for the pre-training process [28].

Masked Language Model (MLM)

In the masked language model objective, tokens from the input are randomly masked by replacing them with a mask token. The objective of the model is then to predict the original token that was replaced based only on its context. This allows for a bidirectional approach where tokens before and after the token in question can be considered [28].

Next Sentence Prediction (NSP)

The NSP is a binary classification task in which the model is trained to distinguish between the original next sentence and a randomly chosen sentence from the corpus [83].

Pre-training is performed on unlabelled data, by definition characterising the training as unsupervised learning [28]. However, Ruder et al. profess that "language modelling is sometimes referred to as self-supervised learning rather than unsupervised learning" based on the fact that it creates its own labels for the pre-training tasks [78, p. 66].

In order to fine-tune the pre-trained model, the BERT model is initialised with the parameters obtained during pre-training. Then, it is fine-tuned with the help of labelled task-specific datasets [28].

Figure 6.4 shows both the pre-training and fine-tuning procedures.

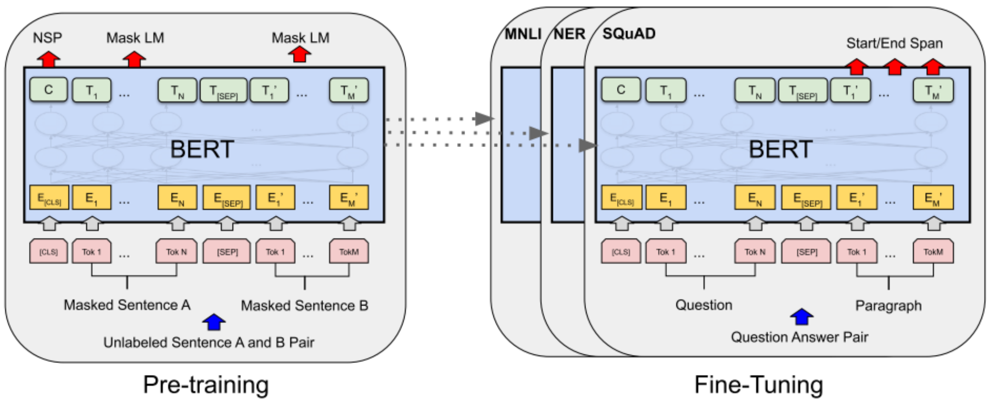


Figure 6.4: Overall pre-training and fine-tuning procedures for BERT [28].

The combination of both pre-training tasks enables BERT to outperform many task-specific architectures on sentence-level and token-level tasks, e.g. sentiment classification and NER respectively [28].

7 RELATED WORK

The previous chapters gave an overview over some developments in the world of deep learning for NLP and NER in terms of technologies. In this chapter, we connect those technologies to quantifiable advancements made with their help and give some historical context.

7.1 NAMED ENTITY RECOGNITION

In order to compare the results of different research groups, we will examine them in regards to an objective benchmark. The *CoNLL03* benchmark containing English and German data was created by Sang et al. in 2003 to evaluate systems on the same data in a competition [80]. The most frequently applied technique tested against the benchmark in the challenge surrounding it was the statistical learning method *maximum entropy model*. Top performances for English and German were F1-scores of 88.76±0.7 and 72.31±1.3 respectively, achieved by ensemble methods with maximum entropy models. Some participants utilised small amounts of unannotated data in addition to the annotated benchmark data. Although they obtained performance gains around 5% in comparison to only using the labelled data with their approach, there was no method that could take advantage of the vast amounts of available raw texts at the time [80].

As an advanced benchmark for German, the *GermEval 2014* NER *Task* was created [7]. It uses an extended tagset with morphologically motivated subtypes and requires prediction of nested NEs, e.g. an entity *person* with the nested entities *first name* and *last name*. The best performance for this challenge with an F1-score of 76.38 was achieved by the ExB group by utilising a combination of techniques and features for their system: conditional random fields (CRFs) and gazetteers, part of speech, tokens, character-level features and word shape [7].

In 2016, Chiu et al. achieved a new state-of-the-art performance on the CoNLL03 benchmark by leveraging a hybrid bidirectional LSTM containing a common convolutional neural network (CNN) architecture with additional pub-licly-available data [18]. Their F1-score was 91.62 on the English part of the benchmark. The trend of utilising large unlabelled corpora continued with Vaswani et al. and their transformer [92]. By adding the bidirectional approach, Devlin et al. were able to create the first pre-trained language model BERT [28]. While they did not achieve state-of-the-art performance on CoNLL03 at the time, trailing 0.29 behind the top result [1], the pre-trained model got applied to many more tasks outside of NER.

A combination of strategies led to the current state of the art for English and German CoNLL03. Wang et al. leverage the automatic concatenation of different contextualised embeddings, as well as traditional non-contextualised embeddings [93]. A controller is tasked with finding the best combination of embeddings, which come from a set of different models, e.g. BERT, multilingual BERT [28], and others. For German-specific embeddings, a BERT model trained on German data was utilised [23]. However, approaches that concatenate multiple embeddings can be computationally costly, because different language models are used as inputs [93]. Other multilingual models include the multilingual BERT called mBERT, and XLM-RoBERTa [20, 28].

Coming back to the GermEval 2014 benchmark, Chan et al. evaluated different transformer architectures and trained two model architectures on German data [14]. Their model *GELECTRA*, based on ELECTRA, outperforms the German model used by Wang et al. [19, 23, 93] and all other common German models [14].

7.2 GERMAN MEDICAL DOMAIN

For German medical texts, the data regulations mentioned in Section 1 increase the difficulty of sharing medical data for research. Several annotation studies created corpora, but those are being kept closed [54]. Because of this, the reported quality of information extraction methods can "neither be evaluated independently nor reproduced externally" [54]. In order to advance research in the German medical domain of NLP, Kittner et al. published the Berlin-Tübingen-Oncology corpus (BRONCO). They also applied state of the art techniques for NER, named entity normalisation, negation and speculation detection, and provided their results as benchmarks [54]. The best results for NER ranged between F1-scores of 77.0 and 91.0 on the validation data, and 72.0 and 90.0 on the test data, depending on the annotation type.

8 SUMMARY

This part of the thesis has introduced the theoretical background of natural language processing (NLP) and the task at hand, named entity recognition (NER). It first offered preliminary knowledge about the most appealing approach, deep learning, in particular of the kind of preprocessing needed to perform it and the challenges faced when working with natural language using the example of tokenisation. Afterwards, it established the basis of deep learning composed of training, supervision, transfer learning, and loss function. How a model can be evaluated was shown by use of the F1 metric and the combination of validation and testing. Finally, the consolidation of both fields led to deep learning for NLP: the encoder-decoder architecture as basis for transformers with their attention mechanism paved the way for the pre-trained language model BERT.

Following these fundamentals, we explored the related work of how they were applied to the task of named entity recognition (NER) in general and in the German medical domain specifically. We found that certain benchmark datasets exist that make a comparison between different types of models possible. The increased difficulty of working with German medical data prompted Kittner et al. to publish the Berlin-Tübingen-Oncology corpus (BRONCO), for which they also published benchmark performances of several model types [54].

In the following part, we go over this BRONCO dataset in detail and perform an exploratory data analysis on it. In order to work with the dataset, we need the hardware and software we present afterwards. On it, we can train the models we introduce and perform hyperparameter optimisation (HPO) on them. Finally, we summarise the experiments we want to conduct in order to achieve good performances on the task of NER.

Part III

SYSTEM

9 DATASET

The dataset discussed in Section 7, published by Kittner et al., is called the Berlin-Tübingen-Oncology corpus (BRONCO) [54]. To our knowledge, it is the only German medical dataset freely available for academic research. Matching its name, the training dataset contains 150 German discharge summaries of cancer patients treated at Charité Universitätsmedizin Berlin or Universitätsklinikum Tübingen. Another 50 are held back by the authors as a test set. The two datasets are called BRONCO150 and BRONCO50 respectively.

In order to publish the dataset in accordance with German law, the authors performed a set of preprocessing steps on the discharge summaries [54]: First, the reports were manually anonymised. Afterwards, they were scrambled at the sentence level, resulting in a set of disconnected sentences. Both methods were applied in the pursuit of making reconstruction of individual reports impossible. As an additional measure to protect the patients' data, researchers wanting to work with the dataset must sign a data usage agreement detailing the terms of usage [54].

The reports were furnished with the necessary information for four different tasks: named entity recognition (NER), named entity normalisation, negation detection, and speculation detection. We focus only on the information needed for NER in this thesis. For this task, annotations with the labels *diagnosis*, *treatment*, and *medication* were created by domain experts [54].

Annotations are given in the form of the CoNLL format, where data files contain one word per line and the end of a sentence is delimited by an empty line [80]. In each line, the word, e.g. *Leber*, is followed by tab-separated features, and its named entity (NE) tag. In the case of BRONCO, only its part of speech (POS) tag, e.g. *NN*, is included as an additional feature. It follows the Stuttgart-Tübingen-TagSet (STTS) tag set, a specific tag set for the German language [81].

The NE tag, e.g. *I-TREAT*, follows the IOB scheme and thus includes the information whether a word is *outside* a NE (O), *inside* a NE (I-XXX), or the beginning of a NE (B-XXX) of type XXX. When two or more entities of the same type are next to each other, the first word of the second entity gets denoted as B-XXX to identify the beginning of another NE [80]. The type is a short form of the label of the word; in the case of BRONCO, this is DIAG for diagnosis, TREAT for treatment, and MED for medication [54].

An example taken from the BRONCO dataset is shown in Table 9.1. It depicts three different cases of the annotation schema. The simplest are one-word annotations like the annotations B-TREAT for "TACE" and O for "Bei". At the beginning of the sentence, we can see the more complex combination of B-TREAT and I-TREAT to delineate a two-word annotation. In the middle of the sentence, we have the case where a one-word annotation with B-DIAG is immediately followed by a two-word annotation with B-DIAG and I-DIAG. It is clear with the use of two B-DIAG that we are looking at two separate NEs.

Word	**POS Tag**	**NE Tag**
MR	NE	B-TREAT
Leber	NN	I-TREAT
:	$,	O
Bei	APPR	O
Zustand	NN	O
nach	APPR	O
TACE	NE	B-TREAT
von	APPR	O
HCC-Herden	NN	B-DIAG
rezidivsuspekte	ADJA	B-DIAG
Laesion	NN	I-DIAG
im	APPRART	O
Segment	NN	O
8	CARD	O
subkapsulaer	ADJD	O
.	$.	O

Table 9.1: BRONCO annotation example with a length of 16 tokens.

9.1 EXPLORATORY DATA ANALYSIS

As described in Section 5.5, the accuracy of the estimated generalisation error of the final model depends largely on the similarity of the distributions of the training, validation, and test datasets. These, in turn, stem from the underlying distribution of the complete dataset. In this section, we conduct an exploratory data analysis to assess the similarity of the three datasets by examining different features, like the amount of tokens in a sentence, and the distribution of NEs. This knowledge also enables making good decisions about the training later on.

9.1.1 *Amount of Tokens in a Sentence*

In Table 9.2, we can see general information about the complete dataset and the splits we performed in order to achieve a similar distribution for all of them. We leveraged a random 70:15:15 percentage split for training, validation, and test, and repeated the process until we found a division that exhibits desirable qualities. Tokens here correspond to word-level tokens, not tokens that are the result of a tokeniser being applied as preprocessing for a model (see Section 5.1).

Dataset	**#Sentences**	**#Tokens**	**Max #Tokens**	**Avg Tokens per S.**
Complete dataset	8972 (100%)	70572	72	7.87
Train subset	6280 (70%)	49091	57	7.82
Val subset	1346 (15%)	10732	72	7.97
Test subset	1346 (15%)	10749	43	7.99

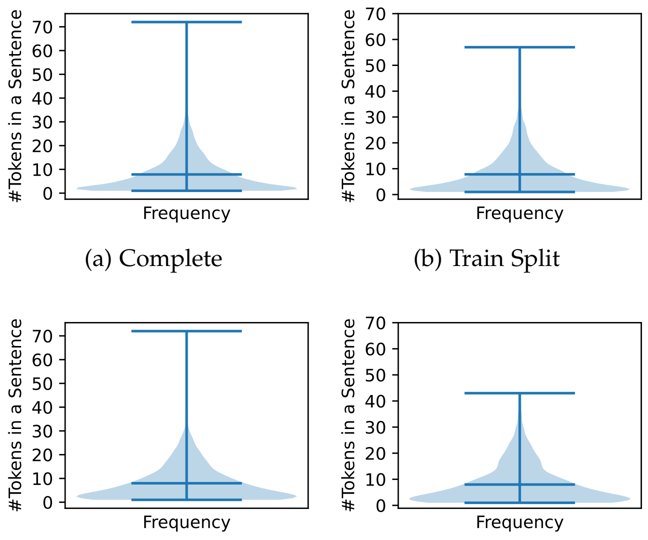
Table 9.2: Preliminary information about the complete dataset in comparison to the three splits for the training, validation, and test datasets.

We can see that the maximum number of tokens in a sentence differs quite dramatically between splits, while the difference of the average number of tokens per sentence is slight. Figure 9.1, however, shows that there are only few sentences with more than 30 tokens in the complete dataset, e.g. only one with 72 tokens. Since the validation subset displays a maximum of 72 tokens, we can tell that the sentence with 72 tokens ended up in the validation dataset. While these differences might influence the training performance and the accuracy of the generalisation error estimate, they are characteristics of the dataset and cannot be compensated completely. Some of these characteristics are presented in Figure 9.1.

The different splits exhibit slight variations of the same distribution, such as the aforementioned maximum number of tokens. This subset is the most balanced one we could achieve without manually distributing the samples and risking introducing additional bias.

From here on, we will focus on the complete dataset, because the splits behave very similarly. Additional figures for the splits of the dataset can be found in Appendix A.

Figure 9.2 zooms in on the lower end of the distribution, i.e. the sentences with 10 tokens or less, of the complete dataset. It is notable that a large number of sentences only consists of one or two tokens. This behaviour is shared throughout all splits. We take this fact into consideration when we discuss the performance of the models in Part iv, especially Section 20.2.



(a) Complete (b) Train Split

(c) Val Split (d) Test Split

Figure 9.1: Distribution of the amount of tokens per sentence, denoted by a line for its minimum, mean, and maximum.

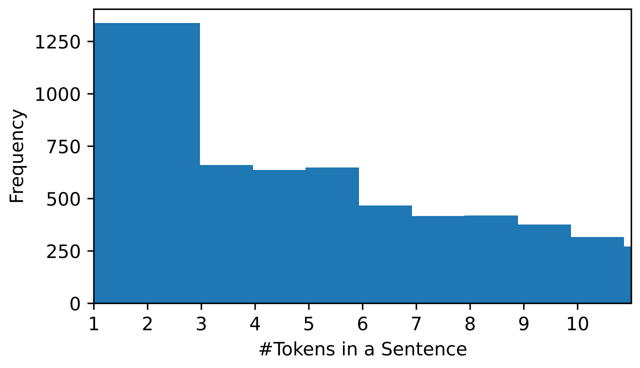
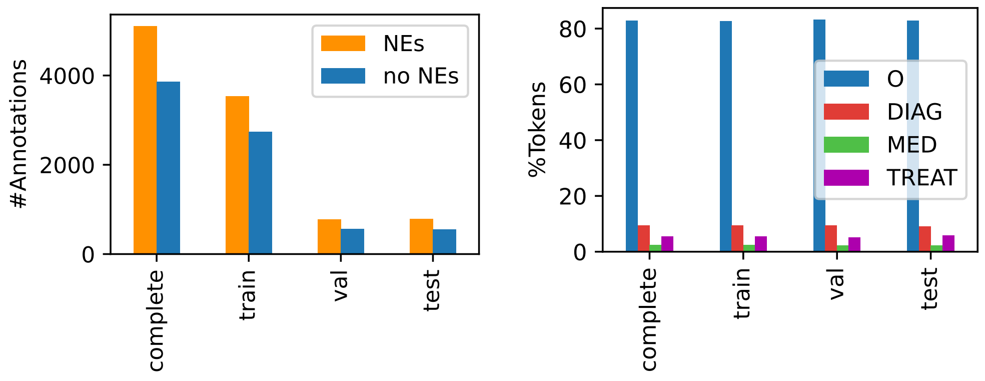


Figure 9.2: Complete dataset: distribution of the amount of tokens per sentence zoomed in.

9.1.2 *Distribution of Named Entities*

Another feature of the dataset is the distribution of NEs. First, we examine how many sentences actually contain NEs. The analysis shows that there are more sentences that do contain one or more NE and fewer that do not, as can be seen in Figure 9.3a.

To examine the distribution of tokens further, Figure 9.3b views the data on a token level instead of the sentence level: less than one quarter of tokens is labelled as a NE, with most of those being diagnosis, followed by treatment, and finally by medication. Figure 9.4 shows the distinction of the NEs into the corresponding NE tags as described in Section 9.



(a) Sentences with and without NEs

(b) Distribution of NEs in the Dataset

Figure 9.3: Occurrence of sentences that have no NE tags versus those that do (a) and general distribution of NE tags in the dataset splits (b).

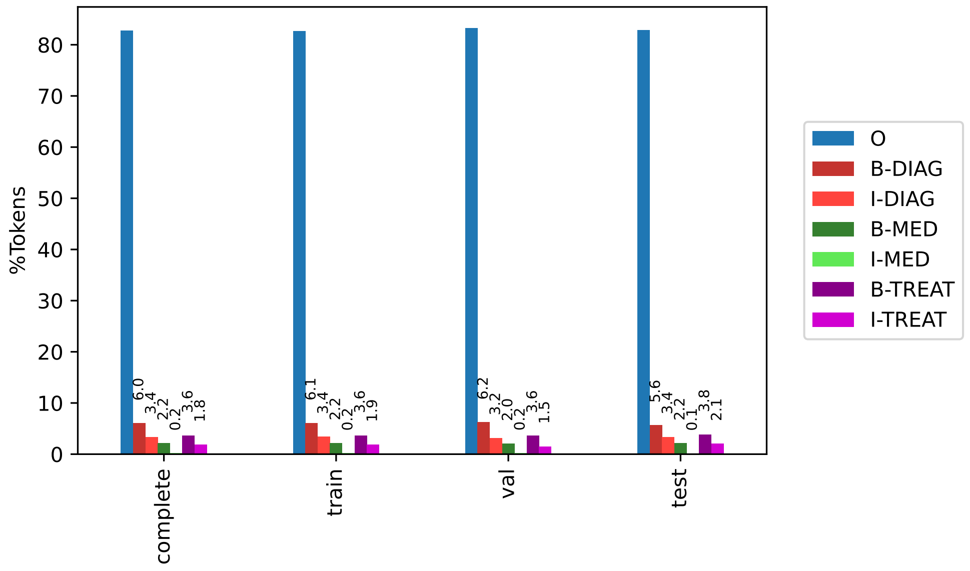


Figure 9.4: Distribution of NE tags in the dataset splits.

9.2 BASELINE RESULTS

As mentioned in Section 7, Kittner et al. published their dataset, the Berlin-Tübingen-Oncology corpus (BRONCO), in order to advance research in the German medical domain of NLP. They applied state of the art techniques for named entity recognition (NER), named entity normalisation, negation and speculation detection, and provided their results as benchmarks.

Table 9.3 shows the results achieved by computing and averaging a 5-fold cross-validation on the BRONCO150 dataset with standard deviation in brackets. F1-Score refers to the micro-averaged F1-score as described in Section 5.4 [54].

Annotation Type	Method	BRONCO150	BRONCO50
Diagnosis	CRF	75.0 (2.0)	**72.0**
	CRF+WE	74.0 (1.0)	71.0
	LSTM	72.0 (1.0)	71.0
	LSTM+WE	**77.0 (8.0)**	**72.0**
Treatment	CRF	82.0 (1.0)	**78.0**
	CRF+WE	81.0 (1.0)	76.0
	LSTM	81.0 (2.0)	76.0
	LSTM+WE	**84.0 (6.0)**	75.0
Medication	CRF	90.0 (0.9)	**90.0**
	CRF+WE	90.0 (0.6)	**90.0**
	LSTM	88.0 (2.0)	89.0
	LSTM+WE	**91.0 (4.0)**	**90.0**

Table 9.3: Validation and test F1-score for baseline methods for NER (CRF and LSTM-CRF), with and without pre-trained word embeddings (WEs) [54].

The authors utilised conditional random fields (CRFs) [59] and a bidirectional long short-term memory (LSTM) network [41] with a final CRF layer (LSTM-CRF) [54]. In combination with those methods, they tested the impact of German (nonbiomedical) word embeddings (WEs) [67]. They learned that the WEs only have a "marginal impact on the CRF, but considerably improve performance of the LSTM-CRF approach" [54].

For the test performance, the models were trained on the available complete BRONCO150 dataset and evaluated on the held back BRONCO50 data (see Table 9.3). The authors attribute the drop for diagnosis and treatment from validation to test results to a possible form of data leakage as result of the 5-fold cross validation. Any data leakage that might have occurred can be disregarded here for the test results. As such, the test results should be considered as more realistic [54].

Kittner et al. mention the possibility of applying "more fine-grained language models" to the dataset in the future, because they were not available at the time. They specifically refer to a domain-specific German language model, and also suggest experimenting with the German instance of the multilingual BERT model [54]. We examine both of these ideas in this thesis.

10 HARDWARE AND SOFTWARE

As mentioned in Chapter 9, a data usage agreement protects the patients and their data. It includes requirements about the security of the hardware the dataset is stored on [54]. To fulfil this agreement, we were offered to work on the high performance cluster *HPC@Charité* as part of a cooperation between the Berlin Institute of Health at Charité – Universitätsmedizin Berlin and SVA System Vertrieb Alexander GmbH [4, 36, 45].

10.1 HIGH PERFORMANCE COMPUTE CLUSTER AT CHARITÉ

On the cluster, we have access to high performance hardware and a software stack allowing users to leverage it. The hardware resources are managed by the *Slurm Workload Manager* in version 22.05.2 [97]. They include [35]: 13 all-purpose CPU nodes with 1408 CPU cores in total, 21 GPU nodes with an NVIDIA A100 40G GPU each, and 2 NVIDIA DGX A100 nodes, each with 8 A100 80G GPUs.

Since benchmarking the model training lies outside the scope of this thesis, we prioritise the allocation of hardware in regards to prompt scheduling instead of runtime consistency. Jobs are then distributed to available nodes with the necessary computing power, and thus are trained on slightly different hardware.

10.2 DEVELOPMENT ENVIRONMENT

As part of the software stack on the cluster, the scientific container software *Singularity* is utilised in version 3.7 to allow developers to "work in reproducible environments of their choosing and design" [57, 58].

We take advantage of the NVIDIA NGC container for *PyTorch* as a base environment [32]. With Singularity's capability of converting *Docker* containers into Singularity containers [66, 84], we create a mutable sandbox container in which we can install packages we need via *Conda*, a manager for packages, dependencies and environments [47], and for rare instances via *Pip*, the package installer for Python [76]. The packages can be found in the provided *requirements.txt*.

We prepare data and train models using the PyTorch integration of the open-source *Transformers* library by *Huggingface* in version 4.21.1 with its associated libraries [94]. Logging is handled by the *MLflow* platform for machine learning lifecycles in version 1.26.0 [17]. For hyperparameter optimisation (HPO), we employ the Huggingface integration of the *Optuna* framework in version 2.10.1 [2].

11 MODELS

We have already examined a pre-trained language model as a concept, namely the BERT model (see Section 6.3). In this section, we introduce the language models we work with to achieve the best results on the BRONCO dataset: GBERT, MedBERT, GELECTRA, mBERt, XLM-RoBERTa, and XLM-RoBERTa GER. We distinguish between two groups of models; monolingual and multilingual models. First, however, we discuss model bias.

11.1 MODEL BIAS

We want to bring attention to the danger of human-like bias in machine learning algorithms, because "learned biases formed on human-related data frequently resemble human-like biases towards race, sex, religion, and many other common forms of discrimination" [34]. Bias correction methods have helped to reduce the effect of harmful learned biases [34]. While an inspection of the biases that might be present in the models in this section lies outside the scope of this thesis, we want to acknowledge their likely existence.

11.2 MONOLINGUAL MODELS

Monolingual models are those models pre-trained on data from a single language. As such, they can in theory be applied to data from other languages, but would be expected to perform poorly.

11.2.1 *GBERT*

The original monolingual BERT model was pre-trained on 16 GB of English data [28, 62].

Since 2019, when Devlin et al. published the original BERT model, the masking objective has been updated: before, a subword token could be masked, leaving the rest of the word's tokens unmasked (see Figure 11.1).

With the introduction of *whole word masking*, it was guaranteed that when masking one subword token all other tokens in the word are also masked out. This change improved performance in their tasks [14, 49].

In addition to adopting the updated masking objective, Chan et al. pre-trained a BERT model in 2020 on 173.4 GB of German data. The result is a monolingual German model called GBERT.

Input:	I am eating.				
Tokens:	I	am	eat	ing	.
Subword Masking:	I	am	MASK	ing	.
Whole Word Masking:	I	am	MASK	MASK	.

Figure 11.1: Artificial example of whole word masking in comparison to subword masking for the word "eating".

11.2.2 *MedBERT*

MedBERT is the only accessible German model pre-trained on data from the medical domain that we could find in our research [6].

As a basis, Becker et al. utilised a German BERT model: before the current GBERT version was created by Chan et al. from Deepset, Deepset had published a prior German BERT version in 2019. They had pre-trained it on 12 GB of German data and evaluated it on a set of different datasets for several tasks [25, 26]. Unfortunately, we are not able to find official information about the pre-training procedure and could not get a reply from Deepset. We assume that the German BERT follows the same procedure as the original monolingual BERT, based on the fact nothing else was published and they were both developed around the same time.

Becker et al. collected a dataset of 67.5 MB of German medical data and further pre-trained German BERT on it [6]. This resulted in a monolingual German model for the medical domain.

11.2.3 *GELECTRA*

Based on the success achieved by BERT, Clark et al. introduced a learning procedure that they called "efficiently learning an encoder that classifies token replacements accurately (ELECTRA)" [19]. The core idea is to replace the MLM objective that masks a token in BERT by a new self-supervised task that supplies a new token instead. This new token is generated by a small generator network (see Figure 11.2). The discriminator learns to distinguish between original and replaced tokens. Generator and discriminator are pre-trained jointly, but only the discriminator is fine-tuned on downstream tasks [19].

The new replaced token detection results in more compute-efficient pre-training and better performance on downstream tasks, because the model learns to distinguish between actual input tokens and those generated by the generator instead of only finding a suitable token in place of a masked one [19].

The monolingual model *GELECTRA* is a version of the English ELECTRA model, pre-trained on 173.4 GB of German data [14].

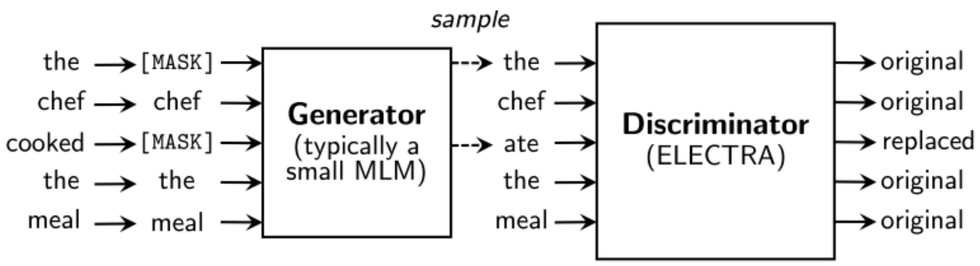


Figure 11.2: Overview of replaced token detection [19].

11.3 MULTILINGUAL MODELS

Multilingual models have been pre-trained on more than one language and tend to be competitive with monolingual models. Especially in cases where only a small amount of training data exists in the target language, *transfer effects* from related languages can benefit the model performance [48].

11.3.1 *mBERT*

At the same time as they published the monolingual BERT, Devlin et al. also developed a multilingual BERT (mBERT) [28, 44]. They had pre-trained it in the same manner as the monolingual BERT, but instead of only utilising one language, they used 104 languages. The decision was based on those languages with the largest *Wikipedia* corpora, as to allow for a sufficient amount of training data. Because the size of the different Wikipedia corpora varied greatly, the authors decided to re-sample the data [27]: By performing exponentially smoothed weighting, they achieved an undersampling of high-resource languages and an oversampling of low-resources languages. This resulted in a more balanced dataset, where the factor by which English would be sampled more than Icelandic improved from 1000 to 100.

Unfortunately, we cannot find official information on the size of the pre-training dataset.

11.3.2 *XLM-RoBERTa German*

The *XLM-RoBERTa* is a cross-lingual language model [21] based on the robustly optimised BERT approach (RoBERTa) published by Facebook in 2019 [62]. The authors of the original monolingual RoBERTa model, Liu et al., claimed that the initial monolingual BERT model was significantly undertrained. They were able to achieve state-of-the-art performance on several challenges by proposing an improved recipe for training BERT models. The modifications include [62]:

1. training for more epochs, with bigger batches, and over more data,
2. removing next sentence prediction objective,
3. training on longer sequences,

4. and dynamically changing the masking pattern applied to the training data.

Liu et al. also added more data to the initial BERT's training dataset of 16 GB, resulting in over 160 GB of training data [62].

Based on this recipe and the approach of scaling the training dataset's size, Conneau et al. from Facebook increased the training dataset to 2.5 TB of filtered *CommonCrawl* data containing 100 languages. This resulted in a large multilingual language model called XLM-RoBERTa [21].

The multilingual XLM-RoBERTa was further fine-tuned on the German part of the CoNLL03 dataset, producing a multilingual language model for token classification specialised on the German language we are referring to as XLM-RoBERTa GER.

11.4 COMPARISON

Table 11.1 shows a summary and comparison of the ways in which the models examined in this section differ. The sizes of their tokenisers' vocabularies can be found in Appendix C.1.

Model	**Pre-Training Procedure**	**Pre-Training Data**	**Domain Pre-Training Data**	**NER Fine-Tuning Data**
BERT	SWM, NSP	16 GB Eng.		
GBERT	WWM, NSP	173 GB Ger.		
MedBERT	*SWM, NSP*	12 GB Ger.	67.5 MB medical Ger.	
GELECTRA	RTD, NSP	173 GB Ger.		
mBERT	SWM, NSP	104 lang.		
XLM-RoBERTa	DM	2.5 TB 100 lang.		
XLM-RoBERTa GER	DM	2.5 TB 100 lang.		CoNLL03 Ger.

Table 11.1: Models and their pre-training procedures, pre-training data, data used for task fine-tuning, and data used for domain fine-tuning. SWM = Subword Masking, NSP = Next Sentence Prediction, WWM = Whole Word Masking, RTD = Replaced Token Detection, DM = Dynamic Mask.

We employ the largest available models for each of the three monolingual and two multilingual models described in this section, i.e. the ones with the most model parameters. The selection of other hyperparameters will be discussed in the next section.

12 HYPERPARAMETER OPTIMISATION

We have already distinguished between model parameters and model hyperparameters with the help of Figure 5.9 in Section 5.5. To summarise: model parameters are *what* the model learns during training and hyperparameters are settings to control its behaviour, i.e. *how* it learns [37, pp. 120–121]. The latter's values can often impact performance considerably [8]. In this section, we present hyperparameter optimisation (HPO) methods that can be employed to choose these hyperparameters and the ranges of values we search in.

12.1 POPULATION BASED TRAINING

Originally, our goal was to utilise the Huggingface integration of *RayTune* for its population based training (PBT) approach, because it is expected to work well with the high number of hyperparameters in deep learning [8, 61, pp. 28–29, 31, 37]. PBT "uses a population of training runs with different settings and applies ideas from evolutionary algorithms" [8, p. 37]. Each member of the population, i.e. model, can exploit another member's information when they perform poorly themselves, e.g. exploit the hyperparameters of a better performing model [87]. Figure 12.1 shows how the training of the population progresses, with exploitation and ion being performed periodically. This ensures that all models in the population perform on a good base level and that new hyperparameters are explored [87].

This process allows to exploit good hyperparameters quickly and dedicate more training time to promising models. In addition to this, hyperparameter values can be adapted throughout training, thus leading to "automatic learning of the best configuration" [87].

Unfortunately, the combination of software and hardware requirements we discussed in Section 10 impeded that strategy: the ray initialisation function fails to detect resources correctly on Slurm [42]. This is likely connected to an issue with object spilling within a Docker image, where data is written on a hard drive if ram is getting too full [3]. While there might exist workarounds for these issues, we decided to prioritise achieving any HPO and switched to another method.

12.2 TREE-STRUCTURED PARZEN ESTIMATOR

As an alternative to PBT, we choose the tree-structured parzen estimator (TPE). It is a single-objective Bayesian optimisation algorithm that is often used for the hyperparameter optimisation of machine learning algorithms [70]. It has been adopted as the standard algorithm of the Optuna framework for hy-

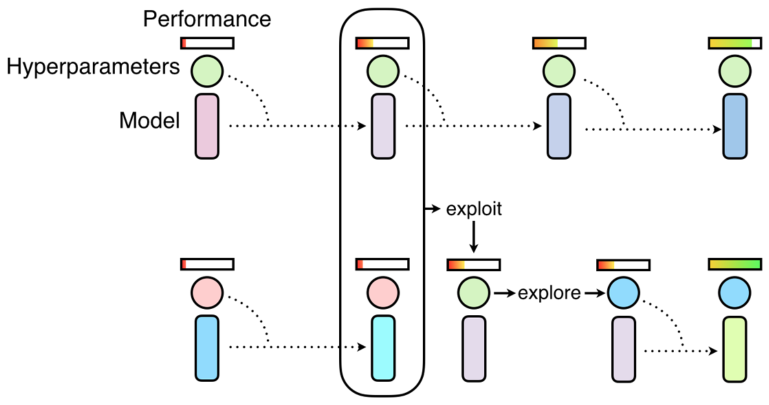


Figure 12.1: Visualisation of population based training (PBT), where information is copied from better performing models (exploit) and then new hyperparameters are explored (explore) [87]. Depicted are two parallel runs, one on top and one on the bottom. In the second step, the bottom model performs badly and is replaced by the better-performing model. On that basis, the hyperparameters are changed to explore more options.

perparameter optimisation, which we leverage for this reason [2]. We choose TPE because it can deal with different types of data and tens of variables instead of only two or three variables like random search or grid search [8, p. 29][70].

12.2.1 *Bayesian Optimisation*

In order to illustrate the advancements made by utilising TPE, we first give an overview over bayesian optimisation (BO), the basis of TPE.

BO is an iterative algorithm that aims to model the mapping of a hyperparameter configuration to the estimated generalisation error for this configuration based on observed performance values via linear or non-linear regression [8, p. 11]. The approximating model is called a *surrogate model* and typically consists of a Gaussian process or a random forest. The following steps are iterated after an initial random choice of configurations has been evaluated and until a termination criterion is reached or the computation budget is exploited [8, p. 12]:

1. fit the surrogate model using all evaluated configurations
2. surrogate model: produce estimates of the performance and the prediction uncertainty for each configuration
3. create a predictive distribution for one test configuration or a joint distribution for a set of configurations
4. establish acquisition function that encodes a trade-off between exploitation and ion and is cheap to evaluate

5. optimise acquisition function
6. generate new configuration candidates for evaluation
7. evaluate new configuration candidates

Figure 12.2 illustrates a snap-shot of the process.

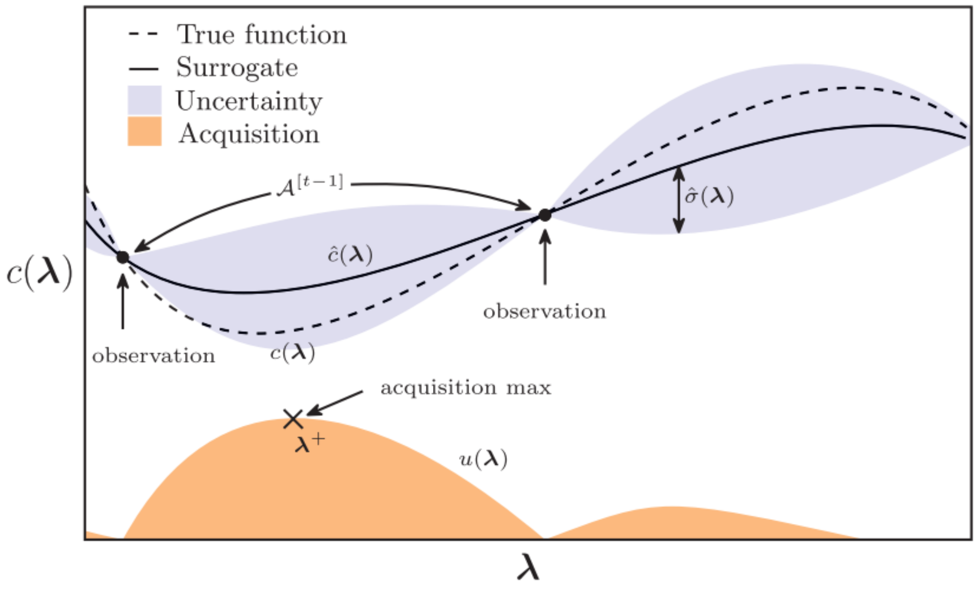

True function
Surrogate
Uncertainty
Acquisition
$A^{[t-1]}$
$\hat{c}(\boldsymbol{\lambda})$
$\hat{\sigma}(\boldsymbol{\lambda})$
$c(\boldsymbol{\lambda})$
observation
observation
$c(\boldsymbol{\lambda})$
acquisition max
$\boldsymbol{\lambda}^+$
$u(\boldsymbol{\lambda})$
$\boldsymbol{\lambda}$

Figure 12.2: Bayesian optimisation (BO) with the goal to minimise the objective function $c(\lambda)$ by use of the surrogate model and its estimated prediction performance $\hat{c}(\lambda)$ and uncertainty $\hat{\sigma}(\lambda)$, as well as the acquisition function $u(\lambda)$, based on all evaluated configurations $A^{[t-1]}$ [8, p. 12].

Which algorithm is used for the surrogate model has great influence on the performance of the optimisation [8, p. 11]. If the hyperparameters to be optimised are real-valued, a Gaussian process regression is used most often. However, it does not support non-numeric or conditional hyperparameters and does not work well in settings with more than ten dimensions, i.e. hyperparameters to be optimised. In addition, their runtime complexity is cubic in the number of evaluated configurations, which can result in significant overhead [8, p. 11].

12.2.2 *Tree-structured adaptive Parzen Estimators as Surrogate Model*

Where Bayesian optimisation utilises Gaussian process regression, TPE uses tree-structured adaptive Parzen estimators [70]. As a surrogate function, they handle continuous variables, as well as discrete, categorical, and conditional variables. In contrast, the Gaussian process regression struggles with discrete, categorical, and conditional variables. TPE also has lower computational complexity and can scale to tens of variables [70].

12.3 PROCESS

In order to utilise TPE instead of PBT, we need to reduce the amount of hyperparameters we want to optimise from hundreds down to tens. Since the set of hyperparameters that have a big influence on performance is often only a small subset of all available hyperparameters, we aim to find those hyperparameters that have the biggest impact [8, p. 9].

We focus on those hyperparameters that have been identified as important by the authors of previous papers in regards to the models to be studied. For GELECTRA for instance, the relevant papers include the paper it was published in [14], the ELECTRA paper for the pre-training procedure [19], and the BERT paper [28] for the underlying architecture.

We identify the learning rate, warm-up ratio for the linear learning rate scheduler, training batch size, number of training epochs, and weight decay as central hyperparameters for optimisation. We define a range of values for each of the chosen hyperparameters that spans all values extricated from the papers. The resulting ranges are denoted in Table 12.1.

Hyper-parameter	**Minimum**	**Maximum**	**Step Size**
Learning Rate	1e-5	5e-4	1e-5
Warm-up Ratio	0.06	0.1	0.01
Batch Size	16	64	16
Training Epochs	2	10	1
Weight Decay	0	0.1	0.01

Table 12.1: Value ranges for hyperparameter optimisation.

Clark et al., who published ELECTRA, mentioned the use of a layer-wise learning rate decay [19]. In addition to a linear learning rate scheduler with a warm-up phase, the learning rate would also be decaying more for each layer for the depth of the network. We could not find an implementation of this functionality in the Huggingface library and only a faulty one by the developers of ELECTRA [46]. The layer-wise learning rate decay proved difficult to implement and its orderly behaviour hard to verify. Thus, we decided against taking this risk and dropped the additional decay.

We leave other hyperparameters at their default values under the assumption that the developers chose them to be well-suited for a variety of tasks.

13 EXPERIMENTS

In order to achieve our goal for this thesis, namely to obtain a well-performing model for the task of NER on BRONCO, we conduct a number of experiments. In this chapter, we explain our rationale behind the types of experiments we choose and how we analyse their results.

In the section about transfer learning (see Section 5.3), we discussed that there are two main ways to adapt models to target tasks, i.e. feature extraction and fine-tuning. Which way should be performed depends on the similarity of the source and target tasks: if they are distant, feature extraction is expected to perform better, whereas fine-tuning should perform better when they are similar [78, p. 77]. A benefit of feature extraction is that it requires less resources. Since we were not able to find any information on how to classify tasks as similar or distant, we compare both approaches for all models.

Because the choice of which hyperparameters training is performed with is central to the model's performance, we want to avoid basing our discussion on non-optimal training runs. For this reason, we perform HPO on both approaches.

Furthermore, we inspect the possibility and feasibility of further pre-training a pre-trained language model on medical data before performing transfer learning on it for the downstream task.

Finally, we inspect the two best models, one monolingual and one multilingual one, in detail and compare their performances on the held back testing dataset BRONCO50 with those achieved by the publishers of the dataset.

In summary, the experiments and analyses we conduct are the following:

1. feature extraction versus fine-tuning
2. feature-extraction HPO
3. fine-tuning HPO
4. feature extraction HPO versus fine-tuning HPO
5. feasibility of further pre-training on domain data
6. inspection of the best monolingual and multilingual model

Unless stated otherwise, we discuss micro-averaged F1-scores on our test subset of the BRONCO150 dataset. Using this version of the F1 metric allows for comparisons to related work [14, 54].

14 SUMMARY

In this part of the thesis, we described the system we utilise to approach the task of NER on German medical data. We discussed the composition of the BRONCO dataset and how we divided it into a training, validation, and test subset while maintaining similar feature distributions.

Then, we presented the high performance cluster HPC@Charité and the development environment we worked with on top of it; a combination of PyTorch, Huggingface, MlFlow, and Optuna makes up the most of our code. We installed them into a Singularity container with the package managers Conda and Pip.

We need this hardware and software stack to train models to perform NER. These models are three monolingual models (GBERT, MedBERT, GELECTRA) and three multilingual models (mBERT, XLM-RoBERTa, XLM-RoBERTa GER). While they all at least partially evolved from the BERT model (see Section 6.3), they differ in their pre-training procedures and the kind and amount of training data. Some were additionally fine-tuned on NER or domain data, i.e. XLM-RoBERTa and MedBERT.

We tried to use a population based training (PBT) approach as a HPO algorithm, but ran into obstacles regarding our software stack. Instead, we chose a Bayesian optimisation approach with tree-structured Parzen estimators as surrogate model. We picked the type of hyperparameter and the ranges in which to optimise them based on the literature relevant to the models we train. We end up optimising the learning rate, warm-up ratio, batch size, training epochs, and weight decay.

Finally, we explained our rationale behind the experiments we conduct and the types of analyses we perform. Those can be found in the next part, where we examine the performances of the models and compare them under different circumstances.

Part IV

RESULTS

15 FEATURE EXTRACTION VERSUS FINE-TUNING

As a first experiment we compare the performance of the two modes of transfer learning mentioned in Section 5.3, i.e. feature extraction and fine-tuning, on the target task. In feature extraction, only the classifier layer is adapted to the task data, whereas the whole model including the classifier is adapted in the fine-tuning approach. We take the hyperparameters for this experiment from the corresponding papers for each model.

We expect the fine-tuning approach to perform better, because it can adjust the model to the data and thus leverage its full potential as a language model. Also, the hyperparameters we use were selected by the authors of the corresponding papers for fine-tuning (see Table 15.1), and thus may give this approach an advantage. Regardless of the approach, we believe that those models that have been trained on either the task or domain data beforehand, i.e. XLM-RoBERTa GER and MedBERT, will perform better than those only trained on the language.

Model	**Learning Rate**	**Learning Rate (Decimal)**	**Batch Size**	**Train Epochs**	**Warm-up Ratio**	**Weight Decay**
GELECTRA	5e-05	0.00005	16	3	0.1	0
GBERT	5e-05	0.00005	16	3	0.1	0.01
MedBERT	5e-05	0.00005	32	3	0.1	0
mBERT	5e-05	0.00005	16	3	0.1	0.01
XLM-RoBERTa	5e-05	0.00005	32	3	0.06	0.1
XLM-RoBERTa German	5e-05	0.00005	32	3	0.06	0.1

Table 15.1: Hyperparameter configurations for each model before optimisation, based on values taken from their corresponding papers.

Figure 15.1 shows the results of training the six models under a feature extraction and a fine-tuning approach. We can see that the mean overall F1-score is around the 80.0 mark for fine-tuning, but only up to approximately 30.0 for the feature extraction approach (see Appendix D.1 for detailed results). We examine the results of each approach further in the next sections.

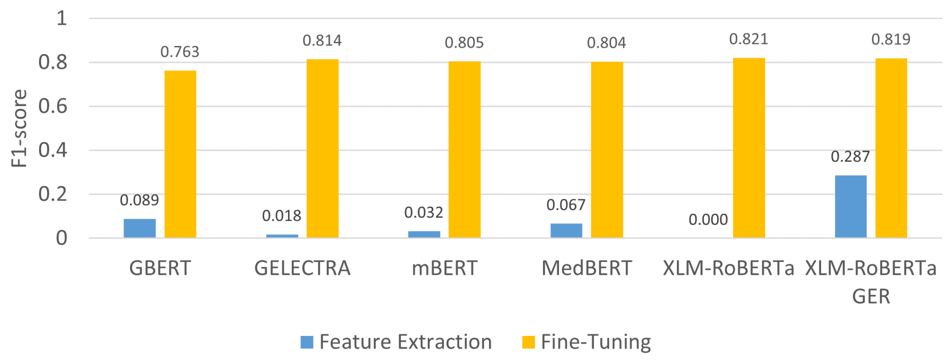


Figure 15.1: Mean overall F1-scores for feature extraction and fine-tuning of three runs per model.

15.1 FINE-TUNING

The best performing models here are GELECTRA, XLM-RoBERTa, and XLM-RoBERTa GER with F1-scores of 81.4, 82.1, and 81.9 respectively (see Figure 15.1).

There is only a difference of approximately 2.0 points between the mean F1-scores of the models in the fine-tuning approach — with the exception of GBERT, where it is a difference of 6.0 to the best performing model XLM-RoBERTa. Figure 15.2 depicts the different performances based on computing three runs for each model.

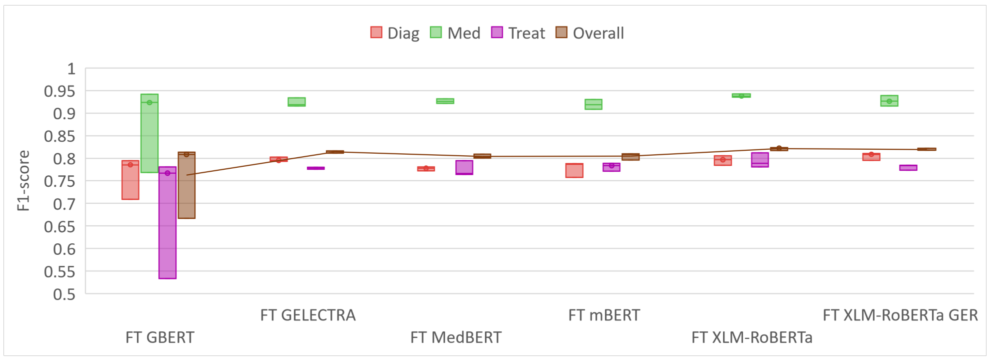


Figure 15.2: Fine-tuning F1-scores of each type of annotations for 3 runs per model, bottom line = lowest run, middle line = middle run, and top line = highest run. The brown line depicts the mean overall F1-score of the models for comparison between the models.

The three types of annotations are depicted alongside the overall F1-score. We will discuss the differences between performances achieved on the types in Section 20.

We see that the difference in overall F1-scores for most models is approximately 1.0, and those for the other types lie around 0.3-3.0. This means that the difference between models is only slightly bigger than the difference within models. The small difference between models could be explained by non-optimal hyperparameter configurations, because most authors only performed a grid search over a small number of values that might not have included the optimal configurations. It is also possible that different models perform differently because some found a more useful representation for the task at hand. We keep this in mind for experiments where the results exhibit stronger effects.

We expected the models trained on domain or task data, i.e. MedBERT and XLM-RoBERTa GER, to perform better based on the knowledge we assumed they would retain from previous training. Because the performances of the models are so similar, we cannot see this effect in the fine-tuning approach.

In terms of the difference between runs of the same model, GBERT constitutes an exception: while two of the three runs perform comparable to the other models, the third run degrades significantly in performance. We repeated this experiment a second time to rule out hardware and software malfunctions, but got similar results. This particular run is also responsible for the lower mean and thus bigger difference to the means of the other models.

We cannot tell if this is a common occurrence for the GBERT model, since the authors of the corresponding paper averaged their measured performance over different downstream tasks, and not the same one as we did. We do, however, find some inconsistencies in the linearity of their training performance that might speak to an underlying issue [14]. Regardless of this inconsistency, there might be other reasons why the training of GBERT is not as stable as that of the other models: the three multilingual models mBERT, XLM-RoBERTa, and XLM-RoBERTa GER might be more stable because they have seen a lot more data in pre-training from which they can extrapolate (see Section 11). The monolingual MedBERT model has seen less data, but this included domain data, which might increase its stability when training in the domain. GELECTRA mainly differs in the pre-training procedure — whole word masking versus replaced token detection, which is developed to be more efficient. This might also influence the stability of training GELECTRA.

15.2 FEATURE EXTRACTION

In the feature extraction approach, we see noticeable differences in the performance of the models. XLM-RoBERTa GER has the best performance with an F1-score of 28.7. The second best is GBERT with 8.9, and the third best is MedBERT with 6.7. Because the actual language model is fixed in this approach, only the classifier that utilises the features is trained. It is the only part that can adjust to the domain data and type of task. Figure 15.3 shows the differences within and between models.

A notable effect is that the previous fine-tuning of XLM-RoBERTa on a German NER task improved its performance significantly from not learning at all to an F1-score of 28.7. Presumably, it gave the language model a chance to adjust to the kind of task in addition to the language. Analogous, the MedBERT language model might have retained and utilised some of the

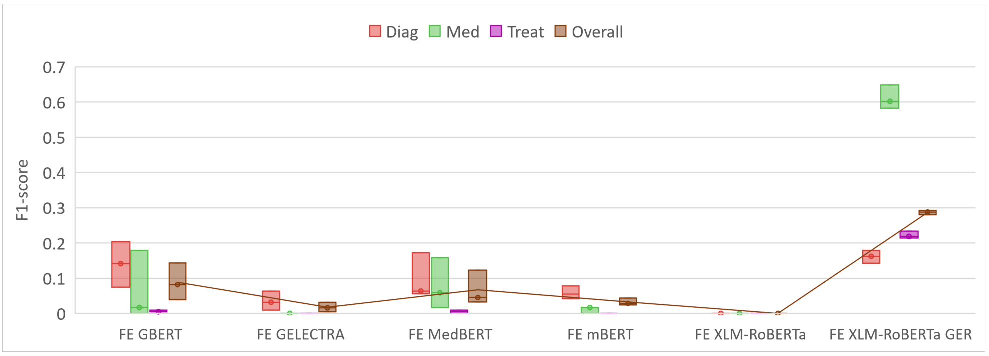


Figure 15.3: Feature extraction F1-scores of each type of annotations for three runs per model, bottom line = lowest run, middle line = middle run, and top line = highest run. The brown line depicts the mean overall F1-score of the models for comparison between the models.

domain knowledge it was pre-trained with. It also displays some instability comparable to GBERT.

Why GBERT performs relatively well in this case is uncertain. It was trained on the same data as GELECTRA and consequently we expected them to perform similarly when the language models are fixed. It is possible that GBERT was able to learn more meaningful representations during pre-training. We would assume that this knowledge would transfer to the fine-tuning approach, which it does not seem to. However, with the large instability we see in GBERT, it is also conceivable that the classifier was initialised with a well-fitting set of random weights for the fixed GBERT model and performed accordingly. It could also be that the hyperparameters chosen for this model were a better fit for the classifier than the other models' hyperparameters and thus enabled it to learn to distinguish better between types. The same can be said for the MedBERT instability and performance, as well as for the difference between all models.

15.3 SUMMARY

This experiment determines that the source and target tasks are similar, based on the fact that the fine-tuning approach performs significantly better for all the models we examine.

Where we can see notable differences between models in the feature extraction approach, those are not present anymore in the fine-tuning approach. It seems that any positive effect that a more sophisticated pre-training might have had on the models got trumped by fine-tuning them on the task data.

However, we chose the hyperparameters for all models based on those parameters given by the authors of the corresponding papers, all of which used them for fine-tuning. To rule out that the hyperparameters gave the fine-tuning approach an unfair advantage, we perform hyperparameter optimisation (HPO) on both approaches in the next sections to compare them accurately.

16 FEATURE EXTRACTION HPO

There are two reasons for why we perform hyperparameter optimisation on the feature extraction approach: firstly, because the hyperparameters we used in the previous chapter were originally chosen for fine-tuning and thus might have underperformed in the setting of feature extraction. Secondly, because those parameters were identified by performing grid search instead of advanced hyperparameter optimisation algorithms. Our goal is to find hyperparameters close to the optimal hyperparameter configurations for our task in this setting and compare the models trained with them to the optimised fine-tuning models later on.

We expect performance to increase significantly, because the choice of hyperparameters has a direct impact on the fitting of the classifier to the dataset. But because only utilising hyperparameters for this is not as powerful as adjusting the full model like in the fine-tuning approach, we believe that even optimised feature extraction is not going to perform as well as unoptimised fine-tuning.

In the following sections, we examine the hyperparameter optimisation of XLM-RoBERTa further as an example before discussing the comparison of all models to their unoptimised performances.

16.1 HYPERPARAMETER RANGES

First, we chose hyperparameter ranges according to the ranges we found in the corresponding papers for each model (see Table 12.1). The best hyperparameter configurations of this optimisation, however, show some concerning effects: The highest possible learning rate we allowed for the HPO is 5e-04, and the maximum number of training epochs is 10. All models were optimised to use this learning rate, and four out of the six models needed the high number of training epochs (see Appendix B.1). Because the ranges were taken from papers dealing with fine-tuning approaches, we assume that they might not include the optimal configurations for feature extraction. This may result in undertrained models. This is why we adjust the hyperparameter ranges as noted in Table 16.1 to allow for more training time and bigger optimisation steps in the form of a higher learning rate.

With these ranges, the best configurations did not exhaust their options and instead stayed well within their limits. They are listed in Table 16.2.

The resulting performances depicted in Figure 16.1 prove that the optimisation with adjusted parameter ranges was necessary to train the models to their full potential under the feature extraction approach. Figures illustrating the difference between runs can be found in Appendix D.

Hyper-parameter	Minimum	Maximum	Step Size
Learning Rate	1e-5	1e-1	1e-5
Warm-up Ratio	0.00	0.1	0.01
Batch Size	16	64	16
Training Epochs	2	20	1
Weight Decay	0	0.1	0.01

Table 16.1: Value ranges for the advanced hyperparameter optimisation.

Model	Learning Rate	Batch Size	Training Epochs	Warm-up Ratio	Weight Decay
GELECTRA	0.06248	64	18	0.04	0.01
GBERT	0.03427	48	16	0.01	0.0
MedBERT	0.02874	32	12	0.01	0.03
mBERT	0.03528	32	16	0.01	0
XLM-RoBERTa	0.03629	32	18	0.02	0
XLM-RoBERTa GER	0.08964	32	16	0.01	0

Table 16.2: Best hyperparameter configuration for each model for the feature extraction approach with adjusted ranges.

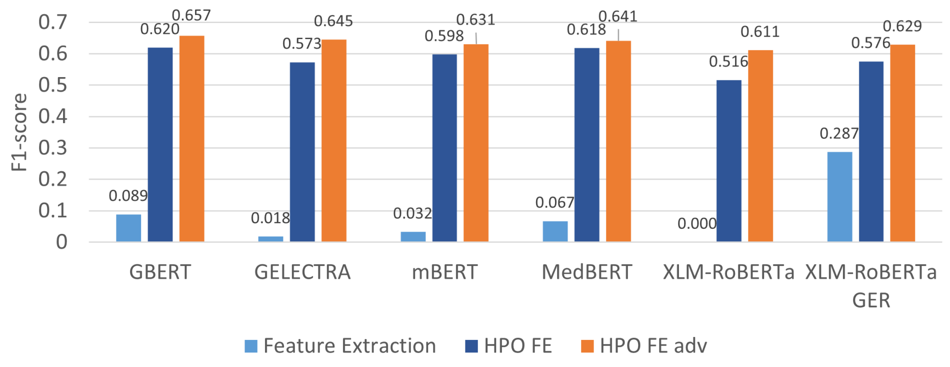


Figure 16.1: F1-scores for feature extraction, the first feature extraction optimisation (HPO FE), and the second advanced feature extraction optimisation with adjusted hyperparameter ranges (HPO FE adv).

The mean improvement from the unoptimised feature extraction approach to the optimised feature extraction increased from 50.0 for the first optimisation to 55.0 for the second optimisation with adjusted ranges. For this reason, we only consider the advanced optimisation with adjusted ranges for future comparisons and analyses.

16.2 HPO PROCESS OF XLM-ROBERTA

The goal of HPO is to find good hyperparameters that facilitate optimising a model by maximising or minimising a certain objective function. In our case, the goal is to maximise the F1-score. We discussed the theory of HPO, and specifically TPE, earlier in this thesis (see Section 12). Now, we explore the application of TPE to the XLM-RoBERTa model by way of example.

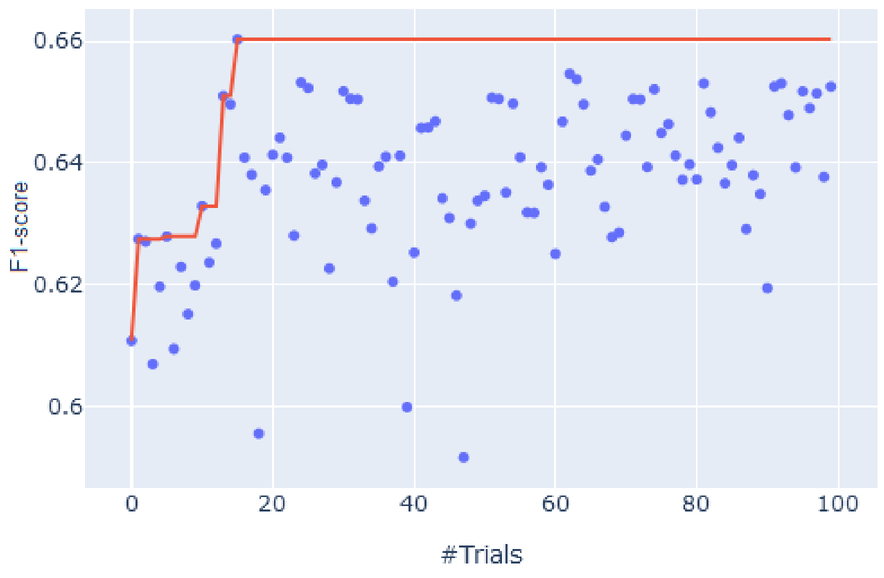


Figure 16.2: Optimisation history of XLM-RoBERTa. Blue points denote trial performance, red line denotes best value so far.

Figure 16.2 depicts the optimisation history of this model. The best configuration was found within the first 20 runs. Subsequent runs were not able to reach that optimisation potential again. Instead, they show a wide dispersion with no visible improvement after 20 runs.

We picked the hyperparameters to be utilised for optimisation based on those that were optimised in the papers corresponding to the models we are leveraging. Figure 16.3 reports the relative importance of those hyperparameters for the optimisation progress of the XLM-RoBERTa model. Those of the other models differ and can be found in Appendix B.3.

We can see that weight decay is the most important parameter of those we consider for optimisation. Also, the number of training epochs and learning rate shows some impact. The batch size and warm-up ratio exhibit barely any significance in the optimisation process.

We can also see this in Figure 16.4, where combinations of hyperparameter values are depicted. The darker the colour of the line is, the better the performance of the resulting model turns out to be. By investigating the weight

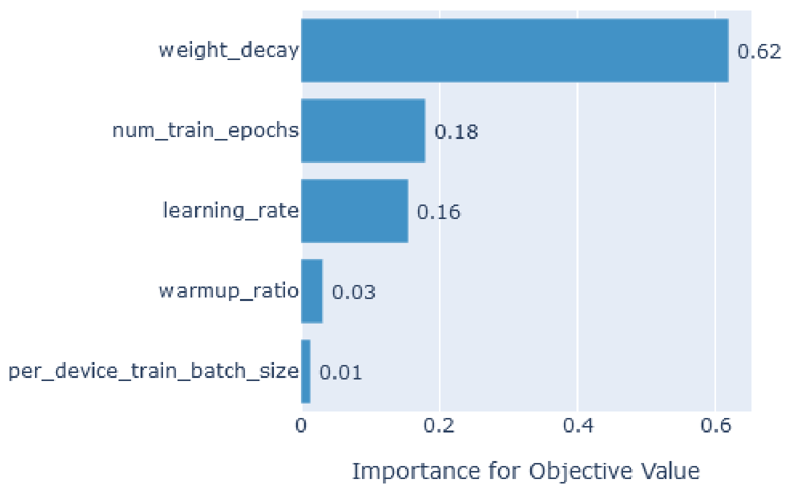


Figure 16.3: Parameter importance of XLM-RoBERTa.

decay, we can see that only lower values result in a good performance. In contrast, the warm-up ratio can have any of the allowed values without influencing the performance of the model significantly. This is all determined by the HPO framework Optuna.

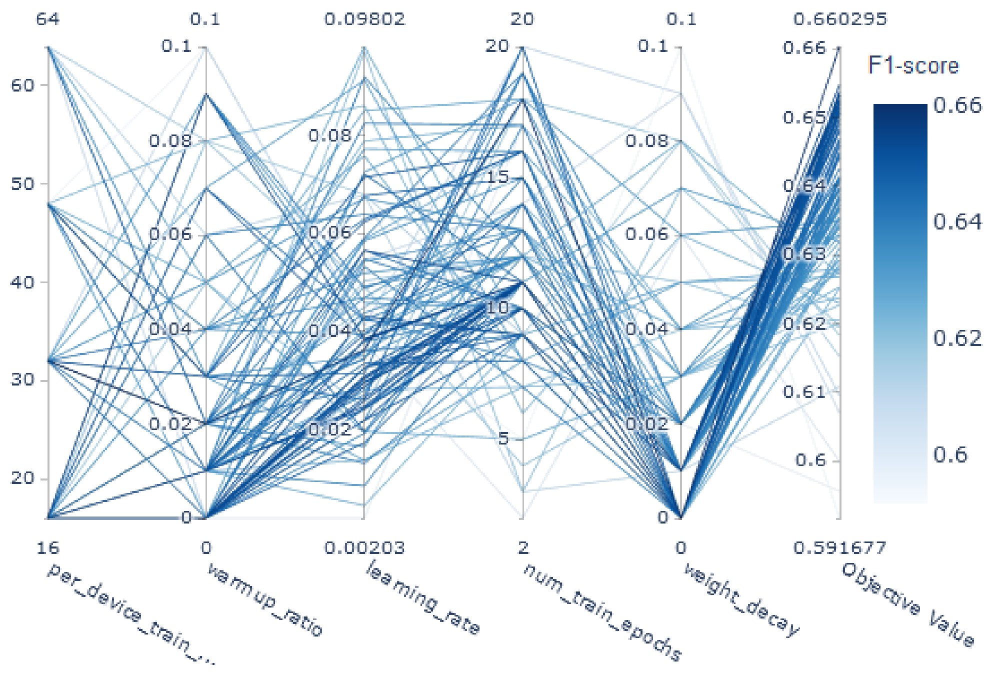


Figure 16.4: Parallel coordinates plot of XLM-RoBERTa. From least important hyperparameter to most important. Objective value is the F1-score, observed hyperparameters are per_device_train_batch_size, warmup_ratio, learning_rate, num_train_epochs, and weight_decay.

The basic idea of TPE is that BO is used to pick increasingly better candidates of hyperparameter configurations to improve the performance, i.e. maximise the F1-score, while exploring different paths of optimisation along the way. Figure 16.5 shows how a good optimisation direction for the weight decay is found within the first 20 trials and how subsequent trials perform better on the basis of this optimisation. There are some outliers where we can assume that a new optimisation direction was tested out and discontinued.

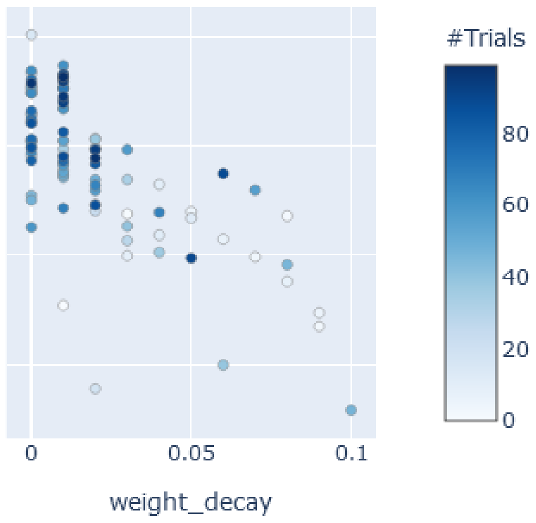


Figure 16.5: Slice plot of the weight decay hyperparameter of XLM-RoBERTa.

This kind of iterative optimisation is performed for all hyperparameters that were chosen for optimisation. While the figures shown in this section only illustrate one example, the optimisations of the other models exhibit similar behaviour.

16.3 OPTIMISED VERSUS UNOPTIMISED

Figure 16.6 displays the feature extraction performance of the six models next to the optimised feature extraction performance and the unoptimised fine-tuning approach's performance. For the optimised feature extraction, the hyperparameters have been adjusted to achieve a better performance on the dataset.

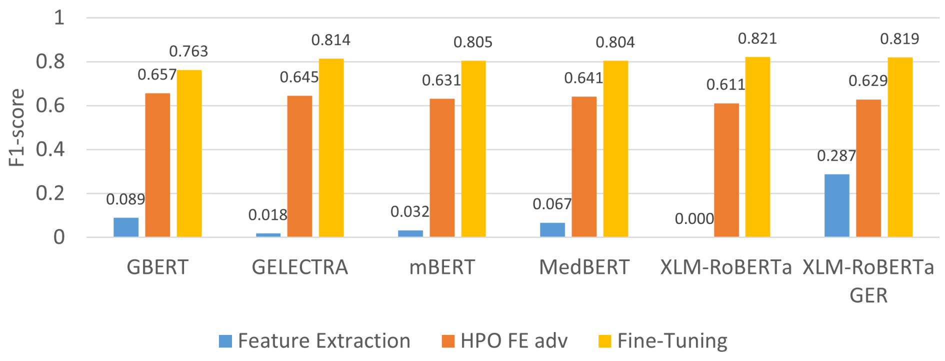


Figure 16.6: F1-scores for feature extraction, HPO of feature extraction, and unoptimised fine-tuning.

As expected, we see a strong increase of the F1-score for the optimised models. This is due to the adjustment of the model to the dataset in the form of the hyperparameters. The increase is not big enough to overtake the fine-tuning approach though, presumably because of the different amounts of hyperparameters; the language model contains a lot more tunable parameters than the classifier, and only that is tuned in the feature extraction approach.

Where XLM-RoBERTa GER had an edge in the feature extraction approach due to the task pre-training, this head start has now been compensated by the other models: all models not based on XLM-RoBERTa performed better than XLM-RoBERTa and XLM-RoBERTa GER. In fact, all three multilingual models performed worse than their monolingual counterparts. The best performances were achieved by GBERT, GELECTRA, and MedBERT, in that order.

We assumed that the difference in performance of the unoptimised feature extraction models was in part due to some hyperparameters working better for the classifier than others. Since these hyperparameters have been adjusted to optimise the objective function, i.e. the F1-score, we can rule this effect out as a factor of why the models perform differently in this experiment. It is very likely that the difference in model performance now is simply based on the way the models represent the input data, and how helpful their specific representation is for the classifier to distinguish between types. We will examine this aspect further for the best two models at the end of our experiments in Section 20.

Regardless of the aspects discussed so far, we would expect the performance of the optimised feature extraction models to degrade when applied to a new or even slightly different dataset. While this is to be expected for every model, we expect it to be more drastic in this case, because the language model is fixed and thus cannot include knowledge from the training dataset, or generalise from it. Thus, it presumably cannot transfer learned knowledge from the domain knowledge to a new dataset. However, because our goal is to find the best performing model for our task, we focus on experiments that aid us in searching for that model instead of using resources on a non-optimal approach.

16.4 SUMMARY

This section gave details about the HPO process performed on all models in the example of the XLM-RoBERTa model. We can identify the weight decay as the most important hyperparameter for this model, followed by the number of training epochs. The best configuration was found within 20 runs, and later runs did not show any improvement. The optimisation with advanced hyperparameter ranges leads to a bigger performance increase than the first ranges taken from the accompanying papers.

We are able to dismiss the idea that the suitability of the models' hyperparameters to the classifier are a main factor in the order of the model' performance. This leaves the features emitted by the language models and their fitness for the distinction of types as central factor of why models achieve different F1-scores.

17 FINE-TUNING HPO

So far, we have achieved the best performances with the fine-tuning approach, but only used the hyperparameter configurations published in the corresponding papers. While we believe that they are a decent fit, we expect some improvements from optimising the hyperparameters for each model.

Table 17.1 denotes the configurations with which the best performances could be achieved.

Model	Learning Rate	Learning Rate (Decimal)	Batch Size	Training Epochs	Warm-up Ratio	Weight Decay
GELECTRA	5e-05	0.00005	48	9	0.07	0.04
GBERT	7e-05	0.00007	48	10	0.07	0.09
MedBERT	6e-05	0.00006	48	10	0.08	0.09
mBERT	9e-05	0.00009	32	8	0.1	0.01
XLM-RoBERTa	4e-05	0.00004	16	10	0.06	0.08
XLM-RoBERTa GER	1e-04	0.0001	48	8	0.06	0.01

Table 17.1: Best hyperparameter configuration for each model for the fine-tuning approach.

We can see that all hyperparameter values stay well within the proposed ranges, except for GBERT, MedBERT, and XLM-RoBERTa and their number of training epochs; we only offered a range up to 10 epochs to be explored in order to keep runtime at a practicable level. This could mean that these models could benefit from training for more epochs. However, the importance of this hyperparameter ranges between 0.02 and 0.04 of 1 and thus can be disregarded in comparison to the overhead a new hyperparameter optimisation would introduce. The importance of different hyperparameters for the models can be found in Appendix B.4.

Figure 17.1 shows the performance differences between the unoptimised and optimised fine-tuning approach. For most models, an increase of 1.0-3.0 points is achieved for the F1-score. The exception is GBERT, where we see a decrease of approximately 11.0 points in the average F1-score over three runs.

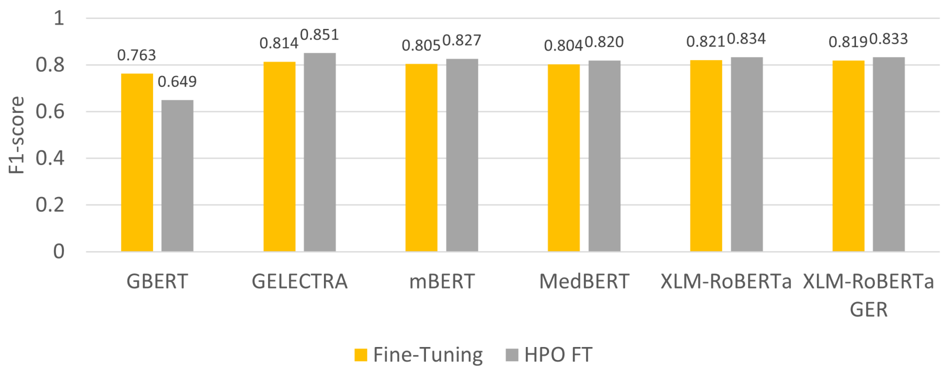


Figure 17.1: F1-scores for fine-tuning and optimised fine-tuning.

With the help of Figure 17.2, it is clear that the decrease in average performance of GBERT is due to the high instability it has already shown in the unoptimised fine-tuning runs (see Figure 15.2). The reasons we discussed in that section are still applicable to the optimised fine-tuning runs (see Section 15).

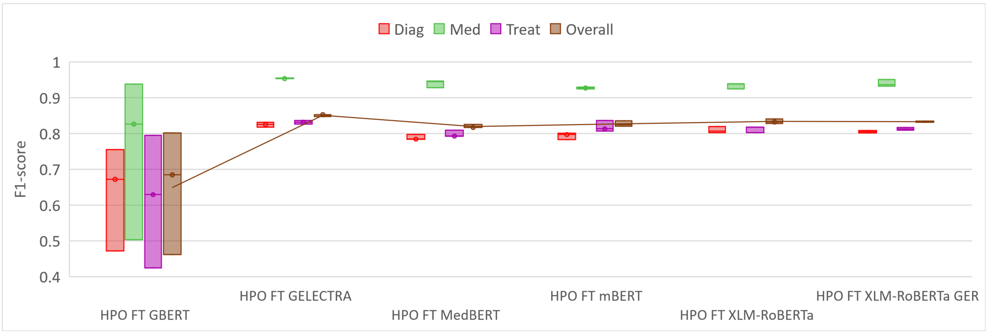


Figure 17.2: Optimised fine-tuning F1-scores of each type of annotations for three runs per model, bottom line = lowest value, middle line = median, and top line = maximum value. The brown line depicts the mean overall F1-score of the models for comparison between the models.

Both Figure 17.1 and 17.2 illustrate the order of the models in terms of average performance as: GELECTRA, XLM-RoBERTa, XLM-RoBERTa GER, mBERT, MedBERT, and finally GBERT. But because the difference to the unoptimised models is small, we check if the performance scores of the different runs overlap (see Table 17.2).

The smallest difference between the best unoptimised and worst optimised models can be found in the case of XLM-RoBERTa and measures 0.4 points. On the other hand, we have GELECTRA where that same difference is 3.0 points. Thus, the performances do not overlap, but show some variety in magnitude. In conclusion, we can say that the HPO does help to achieve a better performance.

GELECTRA and mBERT have benefited more from the HPO than for example XLM-RoBERTa. If this is a theme or just a single occurrence would

Model	Max. Run FT	Min. Run HPO FT	Difference
GELECTRA	81.6	84.7	3.1
MedBERT	80.9	81.7	0.8
mBERT	80.9	82.0	1.1
XLM-RoBERTa	82.4	82.8	0.4
XLM-RoBERTa GER	82.2	83.2	0.9

Table 17.2: Difference between the best unoptimised and worst optimised run for each model.

have to be determined with further experiments. This question, however, lies outside the scope of this thesis.

18 FEATURE EXTRACTION HPO VERSUS FINE-TUNING HPO

In the last sections, we have compared the fine-tuning approach and the feature extraction approach and examined HPO on both. In this section, we combine our insights.

We have seen that the feature extraction models can be improved significantly by performing HPO on them. But even the unoptimised fine-tuning models perform better than the optimised feature extraction models. After optimising the fine-tuning models, we also see performance improvements — with the exception of the unstable GBERT. Figure 18.1 presents a condensed view of these results (see Appendix D.1 for detailed results).

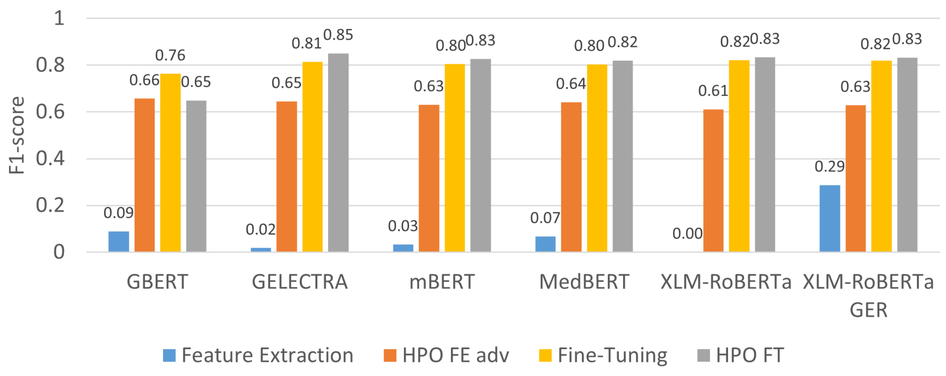


Figure 18.1: F1-scores for feature extraction, the advanced feature extraction HPO, fine-tuning, and optimised fine-tuning.

We believe that this higher increase, however, is only interesting in a research setting: we consider the performances of the models trained under the unoptimised feature extraction approach as too low to be helpful in any realistic setting. While F1-scores around 64.0 as in the optimised feature extraction approach might be useful in some contexts, it is unlikely that a situation exists in which there are no adequate computing resources to leverage the fine-tuning approach, but sufficient resources to perform HPO. Even though the F1-score of the fine-tuning approach increases less, that increase is more relevant, because the small increase of up to approximately 4.0 points directly increases the top performances.

Before we examine the best monolingual model GELECTRA and the best multilingual model XLM-RoBERTa further, there is another avenue of training we want to consider; including domain data in the pre-training procedure. We discuss this in the next section.

19 PRE-TRAINING

We have explored two avenues of transfer learning techniques so far. Both utilised a pre-trained language model as a basis. Where that basis was kept unchanged in the feature extraction approach, it was tuned alongside the classification layer in the fine-tuning approach. For both of these approaches, we were limited in the amount of training data we could train on, because it has to be annotated for the downstream task.

Another avenue whose feasibility we can explore is further pre-training an already pre-trained language model on domain data. For this, unannotated data can be used for the self-supervised pre-training objectives described in Section 6.2. The process is called domain-adaptive pre-training (DAPT) [99].

19.1 CONSIDERATIONS ABOUT DAPT

While further pre-training increases the masked language model prediction accuracy during the pre-training process, it does not always improve performance on downstream tasks [99]. Zhu et al. hypothesise that "further pre-training encodes shallow domain knowledge that has obvious influence only when there are insufficient labelled data providing task-specific knowledge for fine-tuning" [99]. After their experiments, they conclude that a smaller amount of fine-tuning data highlights the importance of pre-training. This means that DAPT is more useful in low-resource environments.

In order to assess whether we are operating in such a low-resource environment, we examine an example of DAPT and decide if we can expect a performance increase that would warrant the additional effort of performing DAPT.

That example is the monolingual model MedBERT. We compare it to its base version, the German BERT model, which did not have DAPT performed on it.

19.2 GERMAN BERT VERSUS MEDBERT

The base model of MedBERT, the German BERT model, has been trained on 12 GB of German data [25, 26]. Then, that German BERT model has been further pre-trained on roughly 70 MB of German medical data, creating the MedBERT model [6].

We consider the question of how much DAPT increased the performance on tasks in the target domain, i.e. medical texts. The authors responsible for the development of MedBERT list the performance for a multilabel code classification task in comparison with the base model German BERT as noted in Table 19.1 [100].

Models	Precision	Recall	F1-score
German BERT	86.04	75.82	80.60
German MedBERT-256	87.41	77.97	82.42
German MedBERT-512	87.75	78.26	82.73

Table 19.1: Comparison of German BERT and MedBERT with a maximum length of either 256 or 512 [100].

We see a maximal difference of 2.13 in the F1-score on this task. We are, however, more interested in the difference in performance on our NER task. Figure 19.1 shows the F1-scores of the fine-tuned MedBERT and German BERT, unoptimised and optimised.

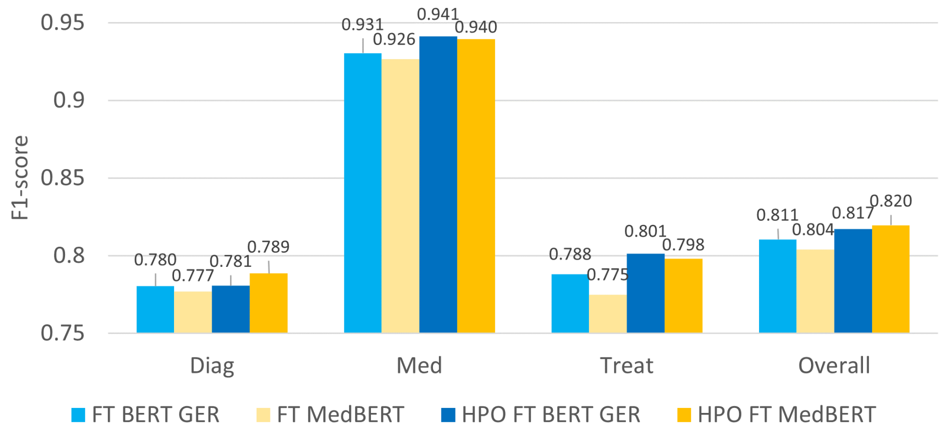


Figure 19.1: Mean F1-scores over three runs for each experiment, for MedBERT and German BERT, unoptimised and optimised, overall and for the three types of annotations.

There exists only a slight difference in performance between the two models over all experiments and annotations: the greatest difference of 1.3 lies between the F1-scores for the treatment annotation of the unoptimised runs. The standard deviations of the three runs of those experiments for that type are 1.0 and 1.7 points, i.e. bigger than the difference of the models. Thus, differences in performance could simply be the expected variations present in a stochastic system.

In conclusion, we do not see a clear indication of whether DAPT could improve the performance of a better performing model like GELECTRA with the help of the available medical data. As a consequence, we decide to focus on using the remainder of this thesis to perform a detailed inspection of the two best models in order to understand why they might behave the way they do.

20 DETAILED INSPECTION

The previous sections determined that the best monolingual model is the optimised fine-tuning approach of GELECTRA and the best multilingual the optimised fine-tuning approach of XLM-RoBERTa. We reached that conclusion by comparing the F1-scores of all models and approaches on our test subset of the BRONCO150 dataset. The next step is to compare our results with those that others have achieved on the BRONCO50 testing dataset.

20.1 PERFORMANCE ON BRONCO50

In order to compare our results to those published by the creators of the BRONCO dataset, Kittner et al., we hand over three trained versions of each GELECTRA and XLM-RoBERTa for evaluation. The models are applied to the held back test dataset BRONCO50 and the F1-score is computed, creating an impartial comparison between researchers.

Figure 20.1 shows the performance of both our models on our test subset of BRONCO150 compared to their performance on the BRONCO50 test dataset.

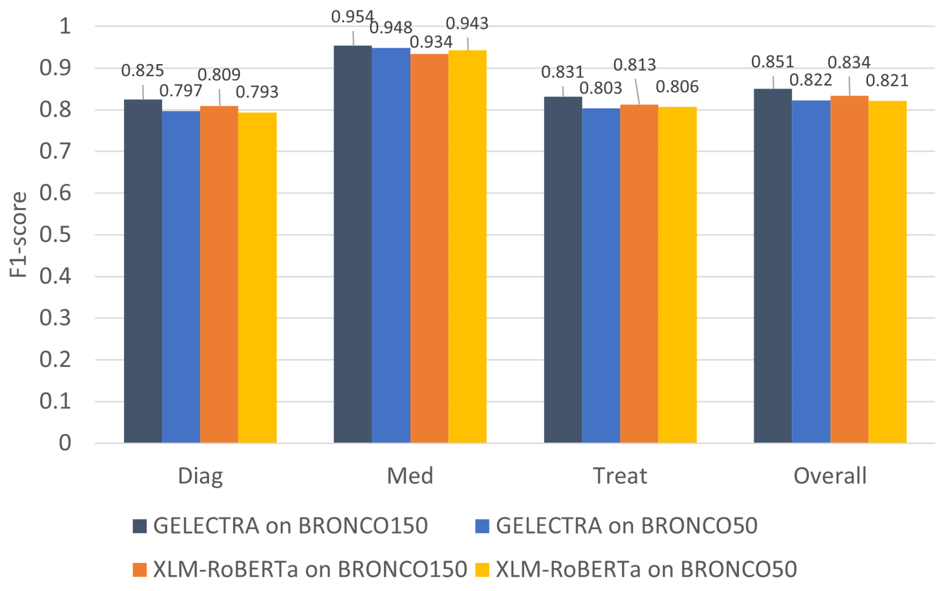


Figure 20.1: Comparison of test results on the test subset of BRONCO150 versus those achieved on the held back BRONCO50 test dataset for the best monolingual and multilingual model. Performances are reported per type and overall.

The results reported for our test subset of BRONCO150 are those already discussed in Section 18; GELECTRA tends to achieve around 1.0 point better than XLM-RoBERTa. Both perform best on the medication type, followed by treatment and, lastly, the diagnosis type with only a very slight drop from treatment of 0.6 points for GELECTRA and 0.4 for XLM-RoBERTa. We will discuss this effect of performing differently depending on the type of NE in more detail in Section 20.2.

The performances on the BRONCO50 test dataset are between 1.0-3.0 points lower. While an inferior performance on test data could indicate an issue of overfitting, a small drop is to be expected because of the datasets. The testing subset of BRONCO150 is taken out of the same documents as the training data and expresses a similar distribution of features (see Section 9.1), whereas the BRONCO50 test dataset is taken from different documents that possibly have slightly different feature distributions. The test subset being taken from the same documents could be considered as a form of data leakage, as already discussed in Section 9.2. For example, one sentence from the same report could be in the training split, and another of the same document in the test split. Those two sentences share the doctor, patient, and the underlying medical information, as well as stylistic features. This is why results on the BRONCO50 testing dataset should be considered as more realistic.

We see only a very slight drop of 1.0 in the F1-score for the medication type, but a bigger one for the treatment type. This means that generalising knowledge about medication is easier for the models than generalising knowledge about treatment. Why that might be the case will be part of our discussion in Section 20.2.

Our main research goal is to further the field of German NLP in the field of medicine by training models that achieve a better performance on downstream tasks. We chose the task of NER on the BRONCO dataset as an example. Figure 20.2 answers the question of if we did indeed achieve a better performance on this task.

Yes, our models outperform those trained by Kittner et al. on all types. Unfortunately, they did not report the overall F1-score and we cannot recreate it since we would need access to the BRONCO50 dataset and their models. The difference in performance ranges from 2.6 points on treatment to 7.7 points on diagnosis, with medication in between with 4.8 points difference.

In the next sections, we investigate the results achieved by the two models in detail. Because we only have access to the BRONCO150 dataset, we perform this analysis on the test subset of BRONCO150.

20.2 TYPES OF NAMED ENTITIES

Throughout our experiments, we have seen that performances vary between the three different types of NEs; diagnosis, medication, and treatment. Generally, results are leading on medication with treatment and diagnosis trailing around 15.0 points behind. Diagnosis and treatment are closer to each other, with F1-scores on treatment usually, but not always, being higher than on diagnosis for the optimised fine-tuning approaches. The higher scores for medication are especially surprising when we consider that the medication type only makes up 2.3% of annotations (see Section 9.1.2). In this section,

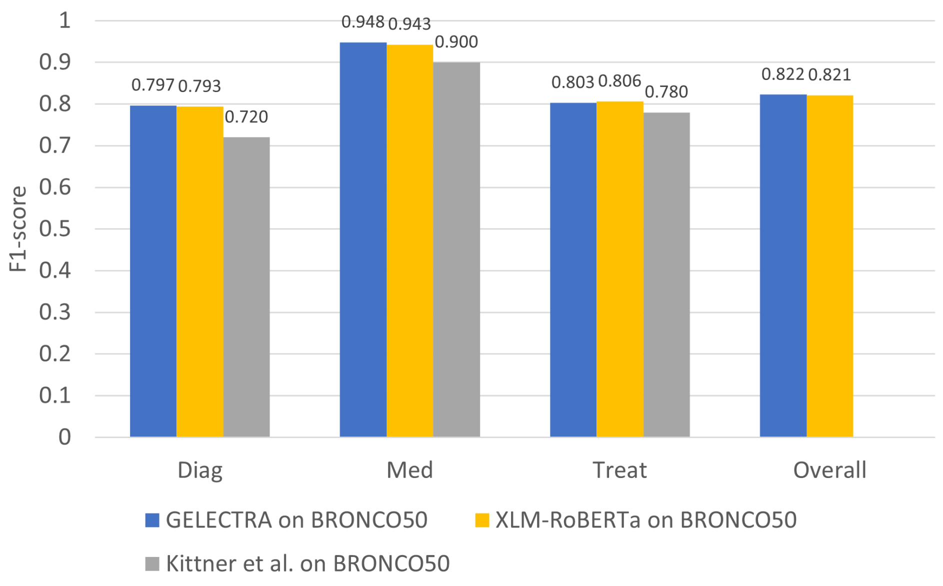


Figure 20.2: Performances achieved by GELECTRA and XLM-RoBERTa on the held back BRONCO50 dataset compared to those achieved by Kittner et al., the publishers of the BRONCO dataset, by use of a CRF.

we explore why the scores might differ by examining the distribution of POS per type and the distribution of NEs per sentence length.

We start with the distribution of NEs per type: Our hypothesis is that NEs of different types might occur in different positions in a sentence and thus in the form of different POS. We test this hypothesis by plotting the distribution of sub-type per universal POS, which are provided as part of the dataset [54, 81]. Figure 20.3 depicts these distributions.

Unsurprisingly, nouns (NN) and proper nouns or names (NE) make up the most occurrences for almost all relevant entities. For treatment, the B-TREAT is dominated by NE, while I-TREAT is spread out over articles (ART), attribute adjectives (ADJA), and prepositions (APPR) among others. For diagnosis, we mainly find attribute adjectives, i.e. ADJA, alongside nouns. Medication also presents as finite modal verb (VVFIN), and non-word (XY) [81].

While these effects might play a part in the slight differences between treatment and diagnosis, they do not seem to be drastic enough to explain the extent of the disparity to the F1-scores on medication.

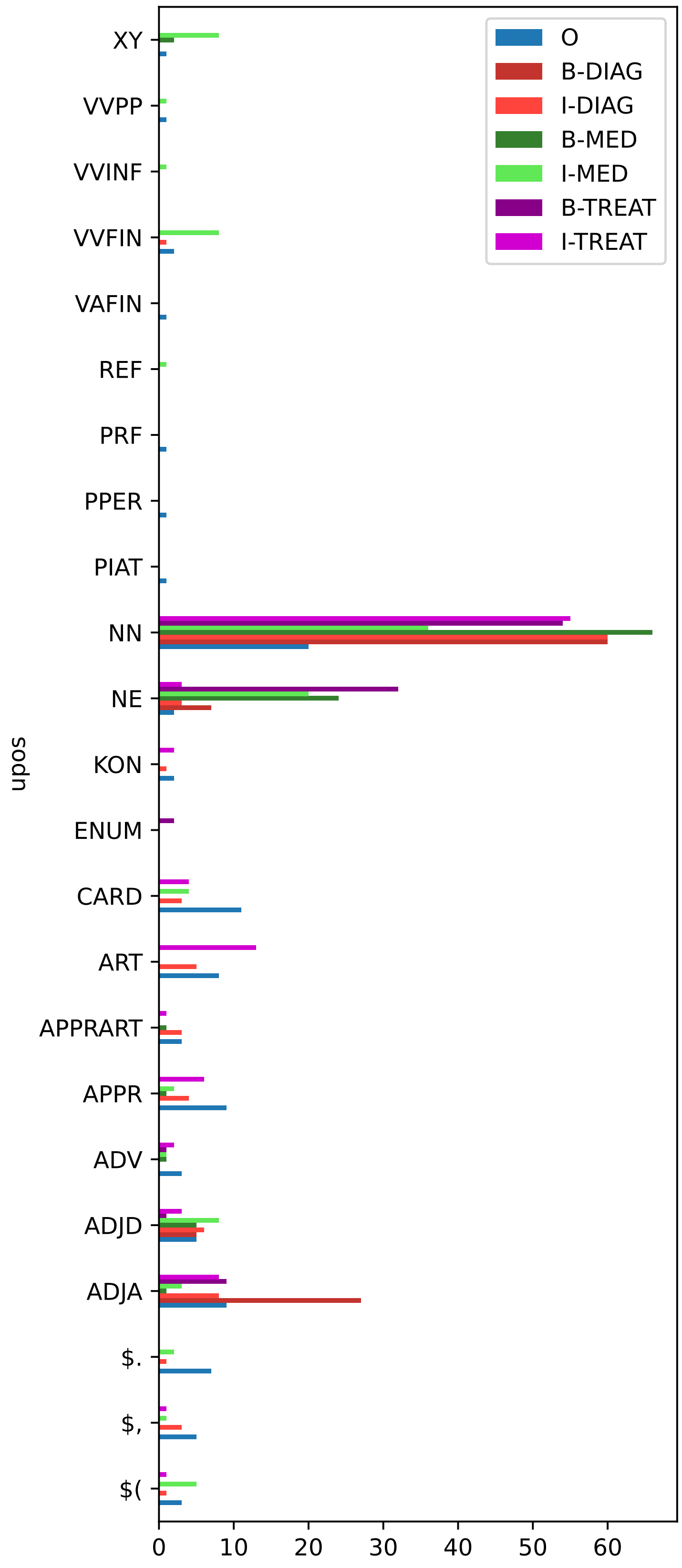
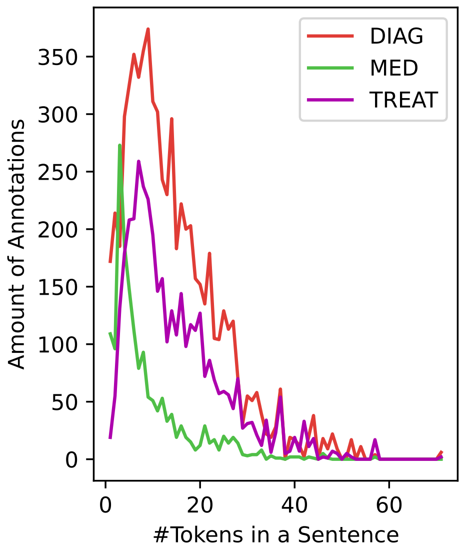
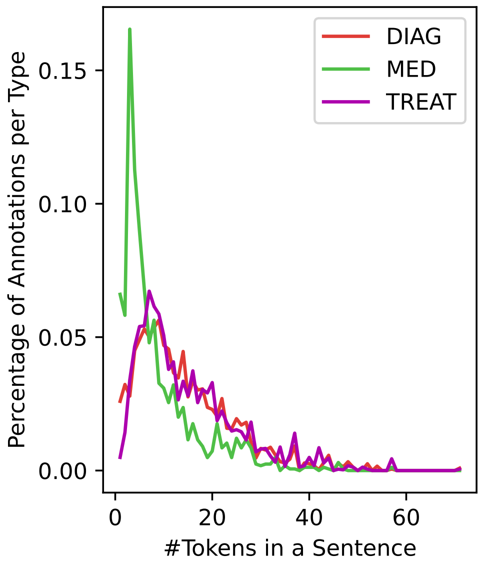


Figure 20.3: Relative distribution of NEs per UPOS in the complete BRONCO150 dataset.

Another interesting question is whether the different types of NEs occur in sentences of differing lengths. Figure 20.4a shows the amount of annotations per type of annotation for the number of tokens in a sentence. Figure 20.4b shows the same data, but as normalised frequency, i.e. percentage, of annotations per type that occur in a sentence with the given amount of tokens.



(a) Absolute



(b) Normalised

Figure 20.4: Absolute (a) and normalised (b) distributions of NEs type per sentence length in the complete BRONCO150 dataset.

The shapes of the distributions of diagnosis and treatment are very similar, with diagnosis annotations occurring more often. Both appear most frequently in sentences with 20 tokens or less, but also appear often in sentences up to 40 tokens in length. Medication, on the other hand, appears most frequently in sentences with 10 tokens or less and barely shows up in sentences longer than 30 tokens.

It is possible that this difference between NE types plays an important part in why it is easier for our models to detect medication NEs. There could, however, be more effects that we have not considered. For example, medication names are often created artificially, which might make them easily recognisable.

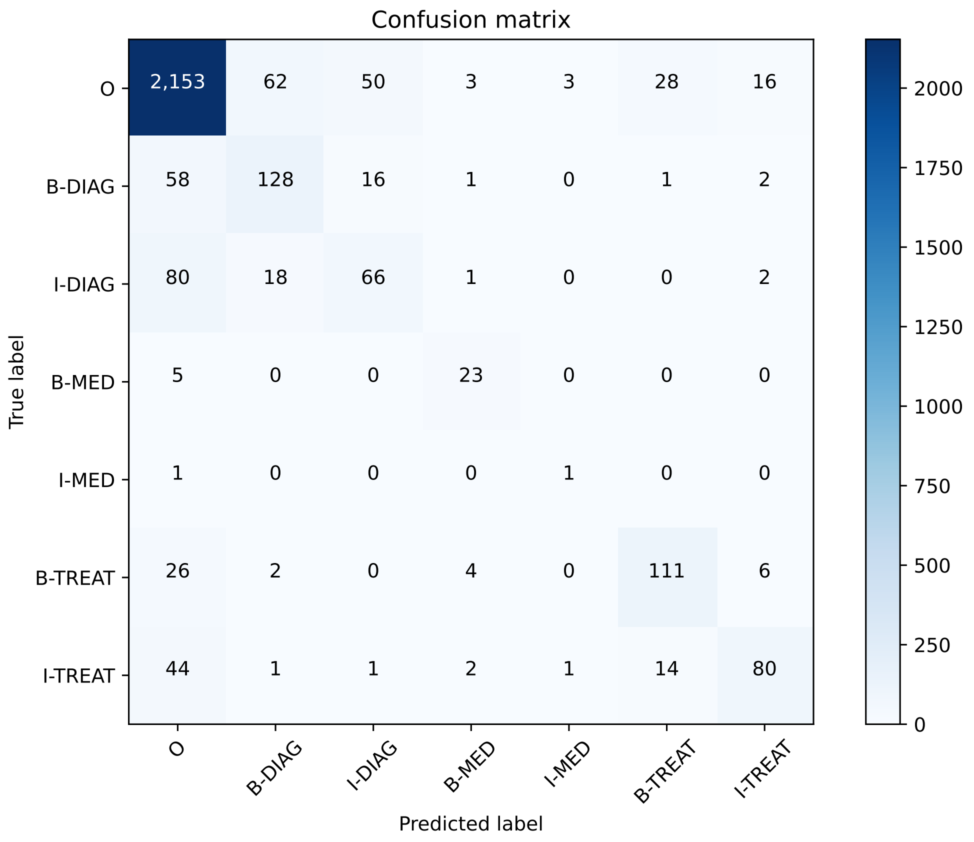
20.3 errors

With an overall F1-score of 85.0 and 83.0 on BRONCO150 for GELECTRA and XLM-RoBERTa respectively, we are interested to see what kind of mistakes were made by the models. For this reason, we examine the errors in terms of two perspectives: first, which NEs were mistaken for which other types and second, which types of errors were made.

20.3.1 *Confusion Matrix*

We are interested in seeing how differently the models behave when they predict NEs. In order to quantify the results, we examine the amount of samples where either one model or both made mistakes. GELECTRA made mistakes in 206 samples in total and XLM-RoBERTa in 224 samples. There is an overlap of 95 in which both models made errors. In 53 of these, they made the same error, and in 42 a different error.

Figure 20.5 shows a confusion matrix made up of the incorrect samples of GELECTRA on the BRONCO150 test split. It depicts whether a token was predicted correctly (diagonal), as FP (top row without diagonal), as a FN (left column without diagonal), or confused for a different NE category (all other cells). The confusion matrix of XLM-RoBERTa can be found in Appendix E.1.



	O	B-DIAG	I-DIAG	B-MED	I-MED	B-TREAT	I-TREAT
O	2,153	62	50	3	3	28	16
B-DIAG	58	128	16	1	0	1	2
I-DIAG	80	18	66	1	0	0	2
B-MED	5	0	0	23	0	0	0
I-MED	1	0	0	0	1	0	0
B-TREAT	26	2	0	4	0	111	6
I-TREAT	44	1	1	2	1	14	80

Figure 20.5: Confusion matrix of incorrect samples of GELECTRA on the BRONCO150 test subset [74].

With 162 FP and 214 FN, more FNs were made. This means that GELECTRA did not recognise the correct entity more often than mistaking a non-entity for a NE. In total, GELECTRA predicted a wrong label for 448 tokens and XLM-RoBERTa for 496. These number are higher than the number of samples with errors, because multiple incorrect token predictions can occur in the same sample.

GELECTRA and XLM-RoBERTa share the general theme, if not the exact values of their confusion matrices. For both, the number of FPs and FNs is higher than the number of confusions between categories. The categories that are most often confused are B-DIAG and I-DIAG, as well as B-TREAT and I-TREAT.

20.3.2 *Types of Errors*

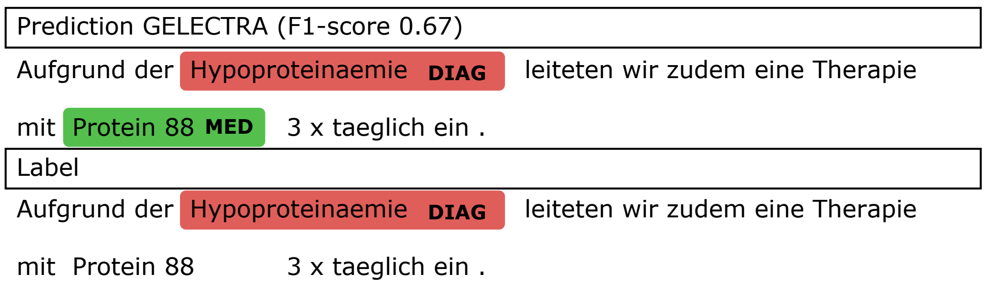
We already started discussing FNs and FPs in the previous sections. In this section, we also discuss error types, but focus on underlying concepts instead of on the number of errors. We present the results of performing a conceptual content analysis on a random choice of examples in the BRONCO150 test split. The following concepts became apparent:

1. simple errors
2. negations, speculation, past
3. additional information
4. symptoms versus diagnosis
5. medication versus treatment

For these concepts, we consider two axes: whether the error is on the side of the label or the prediction, and whether it is a FP or FN. Unfortunately, quantifying these qualitative results lies outside the scope of this thesis. While we interpret the data to the best of our ability, we are not medical professionals and therefore some interpretations might be misleading.

Below, we examine examples for each of the error concept categories.

1. Simple Errors
Humans, as well as models, make mistakes. Some of these mistakes are more obvious than others. Figure 20.6 shows an example where a medication was not labelled but predicted, creating a FP error. We discuss the thought process of why Kittner et al. might not have labelled the NE at the end of this section.


Prediction GELECTRA (F1-score 0.67)

Aufgrund der Hypoproteinaemie **DIAG** leiteten wir zudem eine Therapie mit Protein 88 **MED** 3 x taeglich ein .

Label

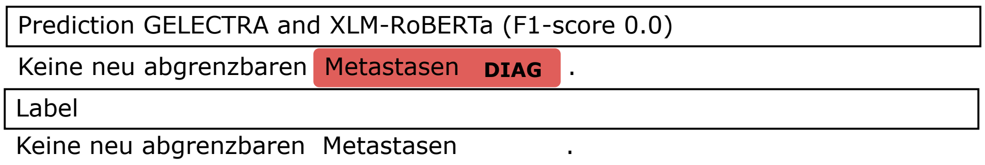
Aufgrund der Hypoproteinaemie **DIAG** leiteten wir zudem eine Therapie mit Protein 88 3 x taeglich ein .

Figure 20.6: Example of a NE of type medication that was not labelled.

Other simple mistakes we discovered in our analysis include labelling or predicting punctuation as part of a NE.

2. Negations, Speculation, Past
Since we are working with natural language, cases are not always clearly cut. Just because a NE is present in the text, it does not always mean that it is currently relevant for the patient. Sometimes, doctors speculate about future developments or recount past progress. They might also exclude certain ideas or note down a lack of something. Whether these concepts are annotated depends on the motivation behind the dataset. Whichever choice is made, it needs to be annotated consistently.

Figure 20.7 illustrates a case where both GELECTRA and XLM created a FP error, because a negated NE was not annotated.


Prediction GELECTRA and XLM-RoBERTa (F1-score 0.0)

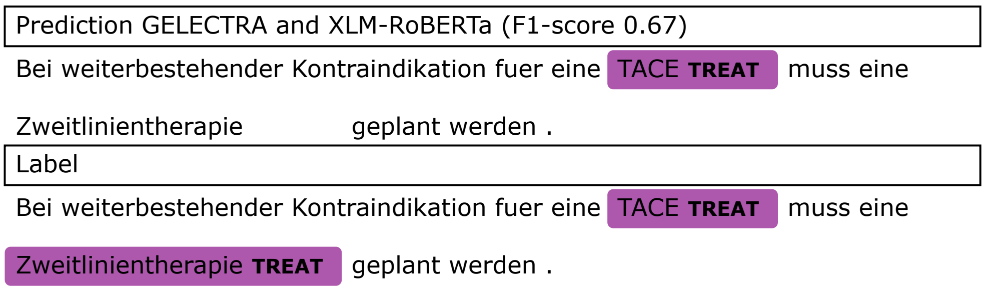
Keine neu abgrenzbaren Metastasen **DIAG** .

Label

Keine neu abgrenzbaren Metastasen .

Figure 20.7: Example of a NE that was not labelled. Assumption: it is not labelled because it is negated.

In contrast, Figure 20.8 shows how a planned procedure is annotated as B-TREAT, but both models predict it to be an O-tag.


Prediction GELECTRA and XLM-RoBERTa (F1-score 0.67)

Bei weiterbestehender Kontraindikation fuer eine TACE **TREAT** muss eine Zweitlinientherapie geplant werden .

Label

Bei weiterbestehender Kontraindikation fuer eine TACE **TREAT** muss eine Zweitlinientherapie **TREAT** geplant werden .

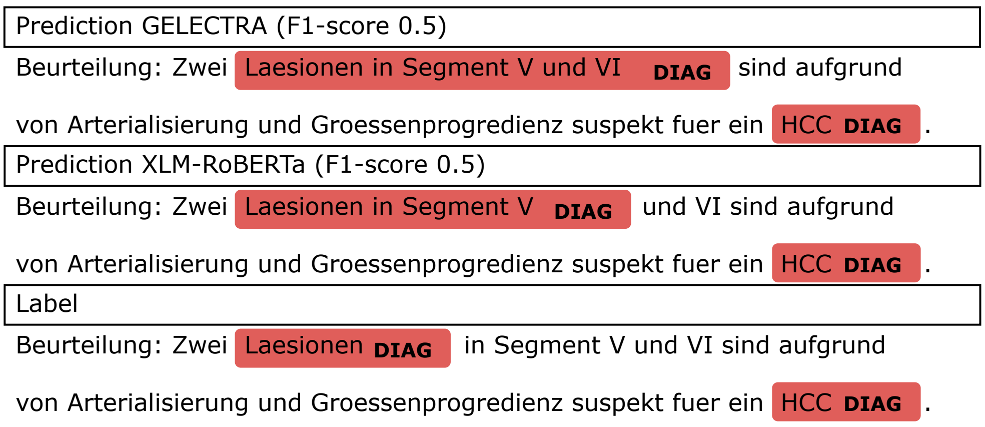
Figure 20.8: Example of a NE that was labelled. Assumption: it is incorrectly labelled because it is only planned.

This might be due to inconsistencies in the annotations, or simply an unrelated error. We discuss the annotation process of the dataset that might influence this type of error at the end of this section.

3. Additional Information
The next case is one where additional information, i.e. locality, is annotated in one example and not annotated in another. In both cases, it leads to an error: in Figure 20.9 it is a FP, and in Figure 20.10 it is a FN.

This demonstrates how inconsistencies can influence the models' decisions. Other information that was inconsistently labelled and therefore predicted includes attribute adjectives, articles, and medication doses.

4. Symptoms versus Diagnosis
Another factor we discovered is the difficult distinction between a symptom and diagnosis. Figure 20.11 shows an example where a type of "Schmerz"



Prediction GELECTRA (F1-score 0.5)

Beurteilung: Zwei Laesionen in Segment V und VI **DIAG** sind aufgrund von Arterialisierung und Groessenprogredienz suspekt fuer ein HCC **DIAG** .

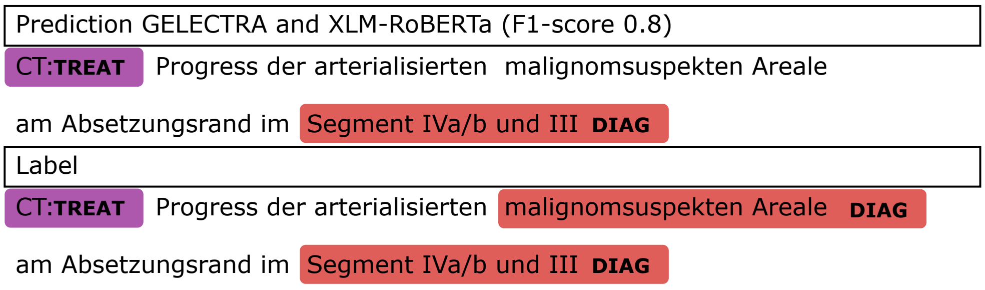
Prediction XLM-RoBERTa (F1-score 0.5)

Beurteilung: Zwei Laesionen in Segment V **DIAG** und VI sind aufgrund von Arterialisierung und Groessenprogredienz suspekt fuer ein HCC **DIAG** .

Label

Beurteilung: Zwei Laesionen **DIAG** in Segment V und VI sind aufgrund von Arterialisierung und Groessenprogredienz suspekt fuer ein HCC **DIAG** .

Figure 20.9: Example of a NE where a locality was not labelled.



Prediction GELECTRA and XLM-RoBERTa (F1-score 0.8)

CT:**TREAT** Progress der arterialisierten malignomsuspekten Areale am Absetzungsrand im Segment IVa/b und III **DIAG**

Label

CT:**TREAT** Progress der arterialisierten malignomsuspekten Areale **DIAG** am Absetzungsrand im Segment IVa/b und III **DIAG**

Figure 20.10: Example of a NE where a locality was labelled.

(pain) is not labelled, but predicted as diagnosis, creating a FP error. This happens because in other samples, different kinds of "Schmerz" (pain) have been labelled as diagnosis. In this example, it seems that it is only a symptom for the diagnosis "Hepatomegalie" (hepatomegaly).

We cannot determine if this is a mistake in the label or if the label is indeed correct.

5. Medication versus Treatment

The same example (see Figure 20.11) also illustrates the messy distinction between medication and treatment. We would argue that treating something with a medication is always also a treatment. As such, we would expect "Prednisolon Therapie" (Prednisolon therapy) to be labelled as a treatment. However, it is only labelled as medication, because the annotation scheme only allows for a single label per word. While this is not an error in the sense of the F1-score, we understand it as an annotation inconsistency that might lead to errors in the sense of the metric in other cases.

20.4 DISCUSSION

We have discussed effects and errors from several different perspectives in these past sections. There are, however, two topics that span over multiple

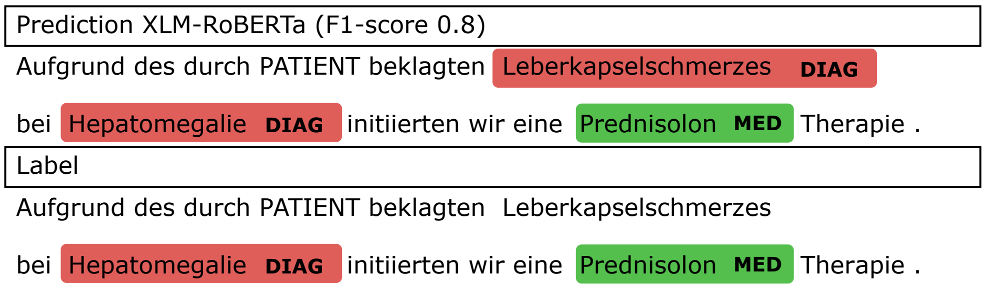

Prediction XLM-RoBERTa (F1-score 0.8)
Aufgrund des durch PATIENT beklagten Leberkapselschmerzes DIAG
bei Hepatomegalie DIAG initiierten wir eine Prednisolon MED Therapie .
Label
Aufgrund des durch PATIENT beklagten Leberkapselschmerzes
bei Hepatomegalie DIAG initiierten wir eine Prednisolon MED Therapie .

Figure 20.11: Example of a NE where a supposed symptom was not labelled and a medication label took priority over a treatment label.

perspectives that we want to expand on here; the difficulty of designing concise and consistent annotation guidelines, and the suitability of the F1-score as metric.

20.4.1 *Annotation Guidelines*

Some of the error concepts we examined so far are rooted in how the authors of the BRONCO dataset decided to annotate and post-process the samples; in the case of multiple accurate annotations, they defined the order of importance as *diagnosis over medication over treatment*. This explains why "Prednisolon" in Figure 20.11 is given the label of a medication instead of a treatment.

In other cases, like planned or negated NEs and additional information, it is unclear to us whether they were supposed to be annotated or not. Examples of both strategies exist in the dataset. This leads to an F1-score that underpredicts the real performance for either strategy.

It is clear to us that defining annotation guidelines that make very clear distinctions in every case is nearly impossible.

20.4.2 *F1-Score*

The micro-averaged F1-score is a common metric for NER. It only counts exact matches as correct instead of partial matches like some other metrics. Table 20.1 illustrates some error calculation examples for the constructed example of "Diabetes Mellitus, behandelt mit Insulin" (Diabetes Mellitus , treated with Insulin).

Even when only part of a multi-word NE is predicted incorrectly, the whole NE receives an F1-score of 0.0. Accuracy on the other hand generally takes partial matches into account, ending up with higher scores, but leads to misinterpretation in cases with a large class imbalance like ours [38, 56]. Whether partial correctness should be rewarded, depends on the task at hand.

Rewarding partial matches would also lead to a higher F1-score in our case, especially considering the types of errors we discussed in Section 20.3.2.

Prediction	**Overall F1**	**DIAG F1**	**MED F1**	**TREAT F1**
B-DIAG, I-DIAG, O, O, O, B-MED	100.0	100.0	100.0	-
B-DIAG, O, O, O, O, B-MED	50.0	0.0	100.0	-
O, I-DIAG, O, O, O, B-MED	50.0	0.0	100.0	-
B-DIAG, I-DIAG, O, B-TREAT, O, B-MED	80.0	100.0	100.0	0.0
B-DIAG, I-DIAG, O, I-TREAT, O, B-MED	80.0	100.0	100.0	0.0
B-DIAG, I-DIAG, O, O, O, B-TREAT	50.0	100.0	0.0	0.0

Table 20.1: Calculation example for the F1-score for the constructed example "Diabetes Mellitus , behandelt mit Insulin" (Diabetes Mellitus , treated with Insulin) with the labels *B-DIAG, I-DIAG, O, O, O, B-MED*.

However, relevant information might get lost in the extraction if only a partial match is predicted. Because we are operating in the medical domain, we therefore believe that it is more advantageous to extract no information for a specific NE than incomplete information. We conclude that the micro-averaged F1-score is a suitable metric for medical NER.

21 SUMMARY

This part presented the results of conducting the experiments described in Section 13:

1. Feature Extraction versus Fine-Tuning
We trained the models with hyperparameters taken from their corresponding papers. Fine-tuning achieved an F1-score of approximately 80.0 for all models, while feature extraction only reached up to around 28.0 and shows notable differences between models. One model, namely GBERT, presented some significant instability in the fine-tuning experiment. The fact that fine-tuning performed better led us to the conclusion that source and target tasks are similar in our case.

2. Feature Extraction Hyperparameter Optimisation
Because the hyperparameters we chose might not have been optimal, we performed HPO on the feature extraction approach twice with different ranges and discussed it in detail. The best configuration was found in less than 20 runs. While both HPO processes increased the performance of the models drastically, the advanced HPO with bigger ranges achieved the best feature extraction scores.

3. Fine-Tuning Hyperparameter Optimisation
The HPO for the fine-tuning approach uncovered the same instability for GBERT as before. Apart from this effect, the HPO on fine-tuning achieved a slight improvement over the unoptimised fine-tuning approach.

4. Optimised Feature Extraction versus Optimised Fine-Tuning
In total, the optimised fine-tuning achieves better F1-scores than the unoptimised fine-tuning approach or any of the feature extraction approaches. GBERT poses an exception to this based on the instability it possesses.

5. Feasibility of further Pre-Training on Domain Data
We explored the feasibility of performing DAPT on a well-performing model by examining its impact on a model that was trained this way. We did not see a clear indication of whether it could improve performance.

6. Inspection of the best Monolingual and Multilingual Model
We examined the best monolingual model, GELECTRA, and the best multilingual model, XLM-RoBERTa, in detail. We applied them to the held back test dataset BRONCO50 and found as expected very similar performances to the F1-scores on our test subset of BRONCO150. We compared our scores to

those published on the BRONCO50 test dataset by the authors and found out that we outperformed their CRF approach.

We looked into why performances on medication tend to be better than on treatment and diagnosis, and identified the distribution of NE types per sentence length as likely cause. Additionally, we identified common error categories as: simple errors, negations or speculations or past, additional information, symptoms versus diagnosis, and medication versus treatment.

Finally, we determined that defining annotation guidelines that do not allow ambiguous annotations are hard to create, and after a brief discussion concluded that the micro-averaged F1-score is a suitable metric for medical NER.

The next part completes this thesis with a summary of its contents, the conclusions we come to, and some ideas for future work in the field of German medical NLP with focus on NER.

Part V

CONCLUSION

22 SUMMARY

In the first part of this thesis, the introduction, we began by explaining the problem and the motivation behind why we work on this task: in order to increase the usability of medical data in the form of natural language for the healthcare system and research, the most important information contained in it needs to be extracted. This can be done by performing named entity recognition (NER) on the data, which is the process of classifying every word in a document as belonging to one of a set of predefined categories. The categories we focussed on are medication, treatment, and diagnosis. We laid down our strategy of first obtaining and preprocessing the data and then choosing models, training and optimising those models on the data in order to finally evaluate and discuss the results.

The next part introduced the fundamentals we need to approach the task at hand. We discussed natural language processing (NLP) and preprocessing on the example of tokenisation, which led into deep learning, the most appealing approach in the NLP domain. We demonstrated how natural language is transformed into numerical values by the use of embeddings and explained how a deep neural network learns with those values as input. Then, we elaborated on the difference between traditional machine learning techniques and transfer learning with its two main ways of feature extraction and fine-tuning. The F1-score as metric gave us a way to measure the quality of the trained model by aggregating precision and recall under the concept of harmonic mean. Following the metric, we explained why we need different splits of the dataset for training, validation, and testing, and which aspects we need to consider when splitting. Then, we moved on to deep learning for NLP, where we illustrated the encoder-decoder architecture and the concept of attention in transformers. The pre-trained language model BERT is the basis for most of the models we chose later on. Subsequently, we connected the technologies from the previous chapters to quantifiable advancements made with their help. Here, we introduced historical context, the benchmark dataset CoNLL03, and the Berlin-Tübingen-Oncology corpus (BRONCO) published by Kittner et al..

The BRONCO dataset is the dataset we decided to train our models on, which is why we examined it in detail. We performed an exploratory data analysis on it, along with discussing the baseline results published by Kittner et al.. In order to switch from theory to praxis in this system part, we set up our development environment on the high performance compute cluster at Charité and described it. On the back of setting up the environment, we chose six pre-trained language models; three monolingual models called GBERT, MedBERT, and GELECTRA, as well as three multilingual models called mBERT, XLM-RoBERTa, and XLM-RoBERTa GER. We reviewed hyperparameter optimisation (HPO) methods, one of which failed to work in our setup, and picked the Bayesian optimisation with tree-structured Parzen estimators as surrogate model. The process we followed is first choosing a set of hyperparameters from papers corresponding to fine-tuning the models and then defining ranges for those hyperparameters that include the values from the papers. The experiments to conduct with the models, the dataset, and the different methods and techniques were defined next.

Part iv examined the results we obtained by conducting the experiments. First, we compared the two transfer learning approaches, i.e. feature extraction and fine-tuning, which resulted in a significantly better performances under the fine-tuning approach. However, because the hyperparameters were taken from the corresponding papers of the models and related only to fine-tuning, we performed HPO on the feature extraction approach. We had to redefine the hyperparameter ranges after a first HPO, and subsequently optimised within bigger ranges. The performance increased dramatically, but not enough to outperform those achieved under the unoptimised fine-tuning approach. The HPO we performed for the fine-tuning approach led to a smaller but more relevant increase, since an increase here means improving the best performing models. We also explored the feasibility of performing domain-adaptive pre-training (DAPT) on a well-performing model by examining its impact on another model. This did not result in a clear indication of whether DAPT could improve performances for our task. Instead, we focused in detail on inspecting the best performing monolingual model GBERT, and the best performing multilingual model XLM-RoBERTa. We compared the performance of those models on the BRONCO50 testing dataset, on which we achieved the best results to our knowledge. We also discussed why different types of NEs lead to different F1-scores, and what kinds of errors were made by the models on the testing subset of BRONCO150.

This part finalises this thesis by recapping its central points in order to discuss the conclusions we came to. Finally, we denote ideas for future work in two branches; ways to improve the results we achieved in this thesis, and what follow-up work could be performed on the back of them.

23 CONCLUSION

In this thesis, we set out to advance research in the field of German medical natural language processing (NLP), focusing on the task of named entity recognition (NER) on the dataset Berlin-Tübingen-Oncology corpus (BRONCO). The last chapter summarised the steps we took in this endeavour. Here, we present the compiled conclusions we gained in the process.

We trained six models on the BRONCO training data, namely GBERT, GELECTRA, MedBERT, mBERT, XLM-RoBERTa, and XLM-RoBERTa GER. When comparing the feature extraction approach and the fine-tuning approach, the second consistently outperformed the first for all models except GBERT. For this model, we experienced instabilities that discourage further use.

Performing HPO increases performances on all accounts — apart, again, from GBERT because of its instability. A higher increase was achieved for the feature extraction approach compared to the fine-tuning approach. We believe that this higher increase, however, is only interesting in a research setting: we consider the performances of the models trained under the unoptimised feature extraction approach as too low to be helpful in any realistic setting. While F1-scores of approximately 64.0, as in the optimised feature extraction approach, might be useful in some contexts, it is unlikely that a situation exists in which there are no adequate computing resources to leverage the fine-tuning approach, but sufficient resources to perform HPO. Even though the increase for the fine-tuning approach is smaller, it is more relevant, because the smaller increase of up to 4.0 points directly increases the top performances.

We were interested in seeing whether DAPT would give MedBERT an advantage over the other models, but could not detect an advantageous effect in our experiments.

Furthermore, we studied MedBERT based on an older and less complex model in comparison to that base model, and still could not find a clear indication for improvement when performing DAPT with around 65 MB of domain data.

Similarly, we expected XLM-RoBERTa GER to outperform XLM-RoBERTa based on the fact that it was fine-tuned on German NER data. We only saw this outcome in the case of the models trained using feature extraction. Based on the fine-tuning approaches, XLM-RoBERTa takes over XLM-RoBERTa GER by small margins of 1.0-2.0 points.

Out of all six models, we identified GELECTRA as the best monolingual, and XLM-RoBERTa GER as the best multilingual model. GELECTRA achieves the highest F1-scores on the held back BRONCO50 testing dataset with an overall F1-score of 82.2. The results of the three types of named entities (NEs) are 94.8 on medication, 80.3 on treatment, and 79.7 on diagnosis. With that, GELECTRA outperforms the performance of the conditional random field (CRF) published as a benchmark by the authors of the dataset, Kittner et al., by 2.3-7.7 points depending on the category.

24 FUTURE WORK

We divide possible future work into two branches: ways to improve the results, and what to do with those results. We start with ideas that could improve the quality of our models or the task of German medical NER.

The BRONCO dataset follows a single label schema, where any word can only be labelled as one NE. This leads to inconsistencies, e.g. between medication and treatment, as discussed. Which label is chosen was defined by the creators of the dataset. We propose a multi-label approach for future datasets, where one word can have multiple labels. This would allow some more flexibility to meet the requirements of an inconsistent language. It would also give future researchers the option to define their own prioritisation based on the needs of a downstream pipeline.

The creation of a bigger dataset for pre-training in the German medical domain would aim in the same direction. Although we did not see a clear improvement from performing DAPT on around 65 MB of German medical data, we expect that to change with a dataset size that is comparable to datasets outside of the medical domain. Analysing how much domain data is needed for this could help researchers to make decisions about their pre-training and fine-tuning regimes in the future. Even just changing the model from one pre-trained on 12 GB of data to one pre-trained on 173 GB could in theory change the outcome.

A similar avenue would be to use medical data from another language to pre-train multilingual models. We would expect different levels of knowledge transfer to German depending on the type of language, because other Germanic languages are similar to German, but Latin and Greek are often the basis of medical terminology.

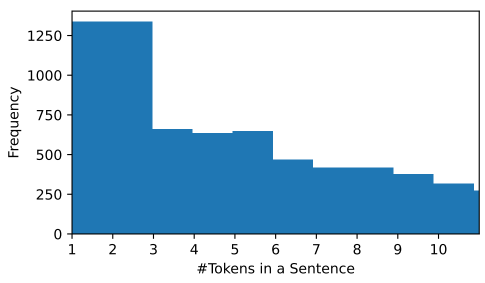
Part of the motivation for this thesis was to make it easier to perform automatic analyses on medical reports. With NER, we worked on the step to extract the most important information from those reports, transforming unstructured into structured data. The next logical step is to link the NEs to medical ontologies for normalisation. In order to do that, a named entity disambiguation model might be needed. It could also be helpful to extract units and doses from medication entities, or to extract other details like location from diagnoses with a rule-based system. A pipeline containing these steps can become increasingly complex, depending on the task at hand. Different kinds of datasets and techniques like deep learning or rule-based approaches could be researched and connected into a functional hybrid system. With systems like that, medical data in the form of natural language can enable the healthcare system and research similarly to the way numerical data is already utilised.

Part VI

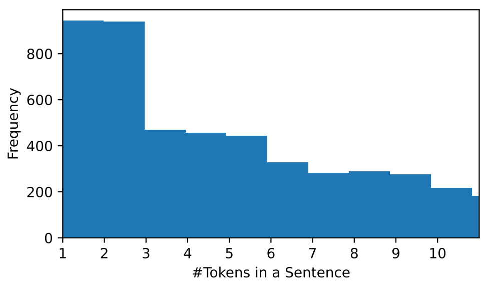
APPENDIX

A

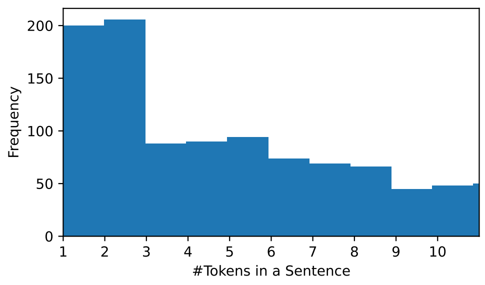
EXPLORATORY DATA ANALYSIS



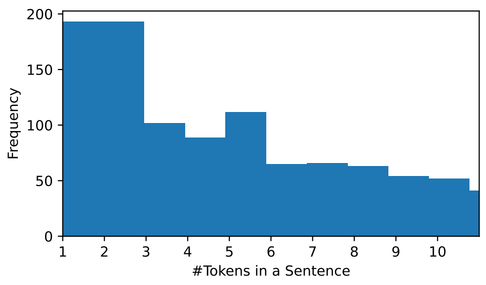
(a) Complete



(b) Train Split



(c) Val Split



(d) Test Split

Figure A.1: Distribution of the amount of tokens per sentence zoomed in.

B HYPERPARAMETER OPTIMISATION

Model	Learning Rate	Learning Rate (Decimal)	Batch Size	Training Epochs	Warm-up Ratio	Weight Decay
GELECTRA	5e-04	0.0005	16	10	0.1	0.0
GBERT	5e-04	0.0005	16	10	0.09	0.02
MedBERT	5e-04	0.0005	16	9	0.08	0.05
mBERT	5e-04	0.0005	16	10	0.06	0.05
XLM-RoBERTa	5e-04	0.0005	16	10	0.09	0.01
XLM-RoBERTa GER	5e-04	0.0005	16	9	0.07	0.09

Table B.1: Best hyperparameter configuration of the first optimisation process for each model for the feature extraction approach.

Model	Learning Rate	Training Epochs	Batch Size	Weight Decay	Warm-up Ratio
GBERT	0.9	0.08	0.01	0.01	0
GELECTRA	0.74	0.17	0.05	0.02	0.01
MedBERT	0.9	0.04	0.02	0.02	0.02
XLM-RoBERTa	0.72	0.22	0.03	0.02	0.01
XLM-RoBERTa GER	0.94	0.02	0.04	0	0
mBERT	0.88	0.08	0.02	0.02	0.01

Table B.2: Relative importance of optimised hyperparameters for the feature extraction approach.

Model	Learning Rate	Training Epochs	Batch Size	Weight Decay	Warm-up Ratio
GBERT	0.14	0.62	0.06	0.12	0.06
GELECTRA	0.17	0.65	0.01	0.07	0.1
MedBERT	0.03	0.89	0.04	0.02	0.02
XLM-RoBERTa	0.16	0.18	0.01	0.62	0.03
XLM-RoBERTa GER	0.23	0.21	0.09	0.37	0.09
mBERT	0.16	0.7	0.01	0.09	0.03

Table B.3: Relative importance of optimised hyperparameters for the feature extraction approach with adjusted ranges.

Model	Learning Rate	Training Epochs	Batch Size	Weight Decay	Warm-up Ratio
GBERT	0.89	0.02	0.01	0.07	0.01
GELECTRA	0.75	0.06	0.05	0.05	0.1
MedBERT	0.95	0.03	0	0.01	0.01
XLM-RoBERTa	0.83	0.04	0.07	0.03	0.04
XLM-RoBERTa GER	0.73	0.11	0.04	0.11	0.01
mBERT	0.49	0.15	0.1	0.17	0.09

Table B.4: Relative importance of optimised hyperparameters for the fine-tuning approach.

C MODELS

Models	**Vocabulary Size**
GBERT	31,102
GELECTRA	31,102
MedBERT	30,000
XLM-RoBERTa	250,002
XLM-RoBERTa GER	250,002
mBERT	119,547

Table C.1: Sizes of the vocabularies of the models' tokenisers.

D RESULTS

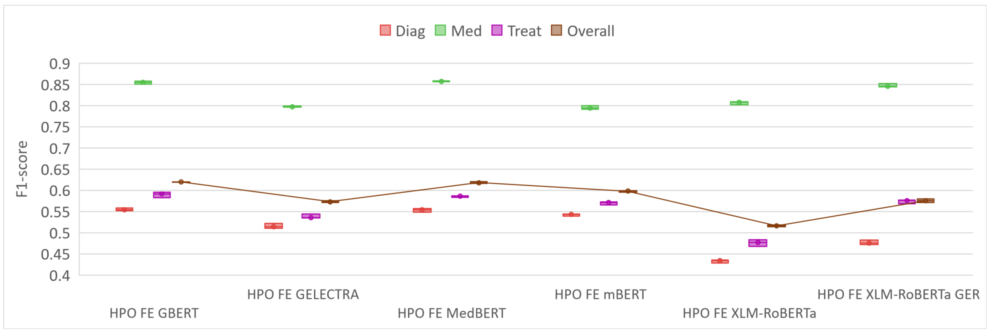


Figure D.1: Optimised feature extraction F1-scores of each type of annotations for three runs per model, bottom line = lowest value, middle line = median, and top line = maximum value. The brown line depicts the mean overall F1-score of the models for comparison between the models.

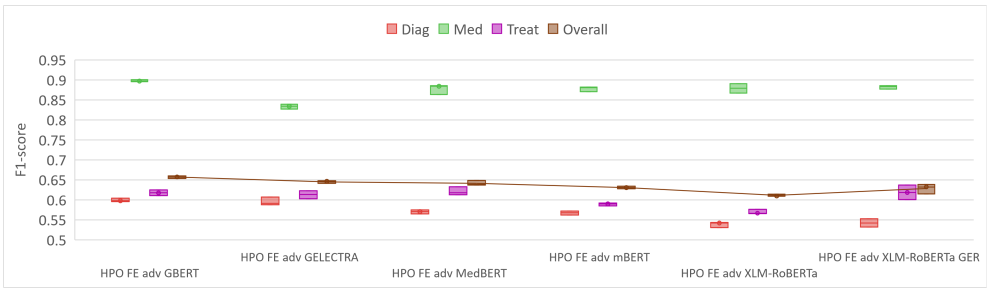


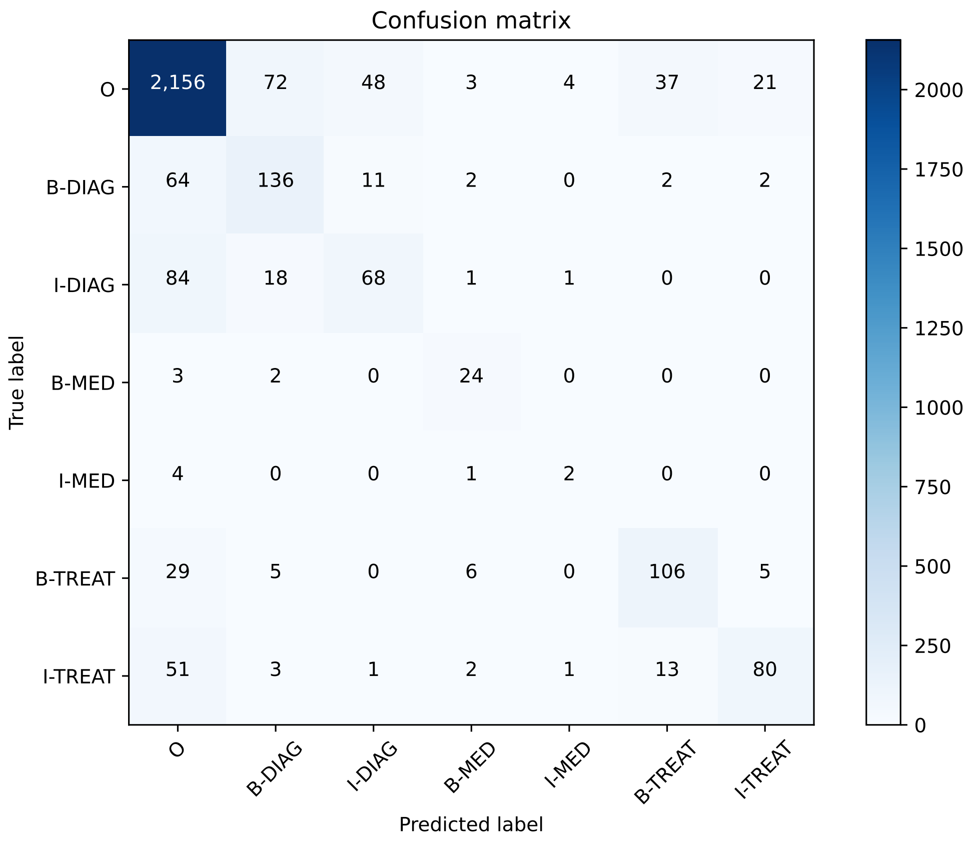
Figure D.2: Optimised advanced feature extraction F1-scores of each type of annotations for three runs per model, bottom line = lowest value, middle line = median, and top line = maximum value. The brown line depicts the mean overall F1-score of the models for comparison between the models.

Model	**Feature Extraction**	**HPO FE**	**HPO FE adv**	**Fine-Tuning**	**HPO FT**
GBERT	8.859394	61.972194	65.70573	76.2870339	64.937448
GELECTRA	1.75741	57.3165679	64.509497	81.431588	85.076951
mBERT	3.24668	59.757641	63.086348	80.4560288	82.688868
MedBERT	6.705451	61.8148989	64.119695	80.3875227	81.961317
XLM-RoBERTa	0.0	51.6056715	61.118938	82.1134781	83.367534
XLM-RoBERTa GER	28.694638	57.5528561	62.865437	81.9364185	83.279413

Table D.1: Detailed F1-scores for feature extraction, optimised feature extraction, advanced optimised feature extraction, fine-tuning, and optimised fine-tuning.

E

DETAILED INSPECTION



	O	B-DIAG	I-DIAG	B-MED	I-MED	B-TREAT	I-TREAT
O	2,156	72	48	3	4	37	21
B-DIAG	64	136	11	2	0	2	2
I-DIAG	84	18	68	1	1	0	0
B-MED	3	2	0	24	0	0	0
I-MED	4	0	0	1	2	0	0
B-TREAT	29	5	0	6	0	106	5
I-TREAT	51	3	1	2	1	13	80

Figure E.1: Confusion matrix of incorrect samples of XLM-RoBERTa on the BRONCO50 testing dataset.

BIBLIOGRAPHY

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. "Contextual string embeddings for sequence labeling." In: *COLING 2018 - 27th International Conference on Computational Linguistics, Proceedings* (2018), pp. 1638–1649. URL: https://aclanthology.org/C18-1139.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. "Optuna: A Next-generation Hyperparameter Optimization Framework." In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019, pp. 2623–2631. ISBN: 9781450362016. DOI: 10.1145/3292500.3330701. arXiv: 1907.10902. URL: https://github.com/pfnet/optuna/.

[3] Yevhen Akimov. *Docker. Using /tmp instead of /dev/shm because /dev/shm has only 31457280000 bytes available - Ray Core - Ray*. 2021. URL: https://discuss.ray.io/t/docker-using-tmp-instead-of-dev-shm-because-dev-shm-has-only-31457280000-bytes-available/1404 (visited on 08/23/2022).

[4] BIH. *Berlin Institute of Health at Charité*. 2022. URL: https://www.bihealth.org/ (visited on 08/18/2022).

[5] BMG. *E-Health – Digitalisierung im Gesundheitswesen - Bundesgesundheitsministerium*. 2021. URL: https://www.bundesgesundheitsministerium.de/e-health-initiative.html (visited on 05/30/2022).

[6] Rolf Becker, Lukas Gilz, and Manjil Shrestha. "Development of a Language model for the medical Domain." PhD thesis. 2021. URL: https://opus4.kobv.de/opus4-rhein-waal/frontdoor/index/index/docId/740.

[7] Darina Benikova, Chris Biemann, Max Kisselew, and Sebastian Padó. "GermEval 2014 Named Entity Recognition Shared Task: Companion Paper." In: *Konvens* 7 (2014), p. 281. URL: https://hildok.bsz-bw.de/frontdoor/index/index/docId/283.

[8] Bernd Bischl et al. "Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges." In: (2021). arXiv: 2107.05847.

[9] Andrew Borthwick. *A maximum entropy approach to named entity recognition*. https://cs.nyu.edu/media/publications/borthwick_andrew.pdf. 1999. (Visited on 06/09/2022).

[10] Tom B Brown et al. "Language models are few-shot learners." In: *Advances in Neural Information Processing Systems*. Vol. 2020-Decem. 2020. arXiv: 2005.14165.

[11] Artem Bugara. *Named Entity Recognition(NER) with SpaCy [with code example] - NewsCatcher*. 2021. URL: https://newscatcherapi.com/blog/named-entity-recognition-with-spacy (visited on 08/04/2022).

[12] Bundesgesundheitsministerium. *Das E-Health-Gesetz | Bundesgesundheitsministerium*. https://www.bundesgesundheitsministerium.de/service/begriffe-von-a-z/e/e-health-gesetz.html. 2017. (Visited on 05/30/2022).

[13] Bundestag. *Patientendaten-Schutz-Gesetz - Bundesgesundheitsministerium*. https://www.bundesgesundheitsministerium.de/patientendaten-schutz-gesetz.html. 2020. (Visited on 05/30/2022).

[14] Branden Chan, Stefan Schweter, and Timo Möller. "German's Next Language Model." In: 2021, pp. 6788–6796. arXiv: 2010.10906. URL: https://github.com/google-research/bert/blob/f39e88/multilingual.md.

[15] Wendy W. Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan. "A simple algorithm for identifying negated findings and diseases in discharge summaries." In: *Journal of Biomedical Informatics* 34.5 (2001), pp. 301–310. ISSN: 15320464. DOI: 10.1006/jbin.2001.1029.

[16] Wendy Webber Chapman, Marcelo Fizman, Brian E. Chapman, and Peter J. Haug. "A comparison of classification algorithms to automatically identify chest X-ray reports that support pneumonia." In: *Journal of Biomedical Informatics* 34.1 (2001), pp. 4–14. ISSN: 15320464. DOI: 10.1006/jbin.2001.1000.

[17] Andrew Chen et al. "Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle." In: *Proceedings of the 4th Workshop on Data Management for End-To-End Machine Learning, DEEM 2020 - In conjunction with the 2020 ACM SIGMOD/PODS Conference*. 2020. ISBN: 9781450380232. DOI: 10.1145/3399579.3399867.

[18] Jason P.C. Chiu and Eric Nichols. "Named Entity Recognition with Bidirectional LSTM-CNNs." In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 357–370. DOI: 10.1162/tacl_a_00104. arXiv: 1511.08308. URL: http://nlp.stanford.edu/projects/glove/.

[19] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators." In: (2020), pp. 1–18. arXiv: 2003.10555.

[20] Alexis Conneau and Kartikay Khandelwal. "Unsupervised Cross-lingual Representation Learning at Scale." In: (2020). arXiv: arXiv:1911.02116v2.

[21] Alexis Conneau and Guillaume Lample. “Cross-lingual language model pretraining.” In: *Advances in Neural Information Processing Systems* 32 (2019). ISSN: 10495258. arXiv: 1901.07291.

[22] Pratap Dangeti. “Statistics for machine learning.” In: Packt, 2017. URL: https://subscription.packtpub.com/book/big-data-and-business-intelligence/9781788295758/8/ch08lvl1sec68/model-building-technique-using-encoder-decoder-architecture.

[23] Dbmdz. *dbmdz/bert-base-german-cased · Hugging Face*. URL: https://huggingface.co/dbmdz/bert-base-german-cased (visited on 07/19/2022).

[24] *Decoder models - Hugging Face Course*. URL: https://huggingface.co/course/chapter1/6 (visited on 08/10/2022).

[25] Deepset. *German BERT | State of the Art Language Model for German NLP*. 2019. URL: https://www.deepset.ai/german-bert (visited on 08/19/2022).

[26] Deepset. *bert-base-german-cased · Hugging Face*. 2019. URL: https://huggingface.co/bert-base-german-cased (visited on 08/19/2022).

[27] Jacob Devlin. *bert/multilingual.md at master · google-research/bert · GitHub*. 2019. URL: https://github.com/google-research/bert/blob/master/multilingual.md (visited on 08/22/2022).

[28] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of deep bidirectional transformers for language understanding.” In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*. Vol. 1. 2019, pp. 4171–4186. ISBN: 9781950737130. arXiv: 1810.04805. URL: https://github.com/tensorflow/tensor2tensor.

[29] Digitales Gesundheitswesen. *Infografik: Chronik der Telematikinfrastruktur - 2001 bis 2020*. 2020. URL: https://digitales-gesundheitswesen.de/chronik/ (visited on 05/30/2022).

[30] Kevin K Dobbin and Richard M Simon. “Optimally splitting cases for training and testing high dimensional classifiers.” In: *BMC Medical Genomics* 4 (2011). ISSN: 17558794. DOI: 10.1186/1755-8794-4-31. URL: http://www.biomedcentral.com/1755-8794/4/31.

[31] *Encoder models - Hugging Face Course*. URL: https://huggingface.co/course/chapter1/5 (visited on 08/10/2022).

[32] Facebook. *PyTorch | NVIDIA NGC*. 2022. URL: https://catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch (visited on 04/07/2022).

[33] Carol Friedman, Lyudmila Shagina, Yves Lussier, and George Hripcsak. "Automated encoding of clinical documents based on natural language processing." In: *Journal of the American Medical Informatics Association* 11.5 (2004), pp. 392–402. ISSN: 10675027. DOI: 10.1197/jamia.M1552.

[34] Daniel James Fuchs. "The Dangers of Human-Like Bias in Machine-Learning Algorithms." In: *Missouri S&T's Peer to Peer* 2.1 (2018), pp. 1–14. URL: https://scholarsmine.mst.edu/peer2peer/vol2/iss1/1/.

[35] GBIT. *HPC@charité - Computing Environments*.

[36] GBIT. *Wissenschafts-IT: Charité – Universitätsmedizin Berlin*. https://tinyurl.com/2jhp3wc3. (Visited on 08/18/2022).

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

[38] Margherita Grandini, Enrico Bagli, and Giorgio Visani. "Metrics for Multi-Class Classification: an Overview." In: (2020), pp. 1–17. arXiv: 2008.05756.

[39] Simon Hessner. *Why are precision, recall, and F1 score equal when using micro averaging in a multi-class problem?* 2018. URL: https://simonhessner.de/why-are-precision-recall-and-f1-score-equal-when-using-micro-averaging-in-a-multi-class-problem/.

[40] Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Atsushi Keyaki, and Naoaki Okazaki. "Joint Optimization of Tokenization and Downstream Model." In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. 2021, pp. 244–255. ISBN: 9781954085541. DOI: 10.18653/v1/2021.findings-acl.21. arXiv: 2105.12410. URL: https://github.com/tatHi/optok4at.

[41] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory." In: *Neural Computation* 9.8 (1997), pp. 1735–1780. ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735.

[42] Florian R. Hölzlwimmer. *ray.init() does not detect local resources correctly on SLURM · Issue #13607 · ray-project/ray · GitHub*. 2021. URL: https://github.com/ray-project/ray/issues/13607 (visited on 08/23/2022).

[43] *How do Transformers work? - Hugging Face Course*. URL: https://huggingface.co/course/chapter1/4?fw=pt (visited on 07/15/2022).

[44] Huggingface. *bert-base-multilingual-cased · Hugging Face*. 2019. URL: https://huggingface.co/bert-base-multilingual-cased (visited on 08/22/2022).

[45] *Ihr führendes IT-Systemhaus in Deutschland | sva.de*. URL: https://www.sva.de/de (visited on 11/02/2022).

[46] Importpandas. *The implementation of layerwise learning rate decay · Issue #51 · google-research/electra · GitHub*. 2020. URL: https://github.com/google-research/electra/issues/51 (visited on 09/09/2022).

[47] Anaconda Inc. *Anaconda Software Distribution*. 2020. URL: https://docs.anaconda.com/.

[48] Tim Isbister, Fredrik Carlsson, and Magnus Sahlgren. "Should we Stop Training More Monolingual Models, and Simply Use Machine Translation Instead?" In: (2021). arXiv: 2104.10441.

[49] Jacobdevlin-google. *Adding Whole Word Masking · google-research/bert@0fce551 · GitHub*. 2019. URL: https://github.com/google-research/bert/commit/0fce551 (visited on 08/18/2022).

[50] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. http://www.cs.colorado.edu/~martin/slp.html. 2020.

[51] Uday Kamath, John Liu, and James Whitaker. *Deep Learning for Natural Language Processing (NLP) and Speech Recognition*. 2019. ISBN: 9783030145958, 978-3-030-14595-8. URL: https://doi.org/10.1007/978-3-030-14596-5.

[52] Shachar Kaufman, Saharon Rosset, and Claudia Perlich. "Leakage in data mining: Formulation, detection, and avoidance." In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* October (2011), pp. 556–563. DOI: 10.1145/2020408.2020496.

[53] Harrison Kinsley and Daniel Kukieła. "Neural Networks from Scratch in Python." In: (2020), p. 658.

[54] Madeleine Kittner et al. "Annotation and initial evaluation of a large annotated German oncological corpus." In: *JAMIA Open* 4.2 (2021), pp. 1–9. ISSN: 2574-2531. DOI: 10.1093/jamiaopen/ooab025.

[55] Hyoun Joong Kong. *Managing unstructured big data in healthcare system*. 2019. DOI: 10.4258/hir.2019.25.1.1. URL: /pmc/articles/PMC6372467/https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6372467/.

[56] Joos Korstanje. *The F1 score | Towards Data Science*. 2021. URL: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6 (visited on 11/14/2022).

[57] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. "Singularity: Scientific containers for mobility of compute." In: *PLOS ONE* 12.5 (2017), e0177459. ISSN: 1932-6203. DOI: 10.1371/JOURNAL.PONE.0177459.

[58] Gregory M. Kurtzer et al. "hpcng/singularity: Singularity 3.7.0." In: (2020). DOI: 10.5281/ZENODO.4289037.

[59] John Lafferty, Andrew Mccallum, and Fernando Pereira. "Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data." In: 2001.June (1999), pp. 282–289.

[60] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. "A Survey on Deep Learning for Named Entity Recognition." In: *IEEE Transactions on Knowledge and Data Engineering* (2020), pp. 1–1. ISSN: 1041-4347. DOI: 10.1109/tkde.2020.2981314. arXiv: 1812.09449.

[61] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. "Tune: A Research Platform for Distributed Model Selection and Training." In: (2018). arXiv: 1807.05118.

[62] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." In: (2019). ISSN: 2331-8422. arXiv: 1907.11692. URL: https://github.com/pytorch/fairseq.

[63] Christopher D. Manning. *Introduction to Information Retrieval*. Cambridge University Press, 2008, p. 506. ISBN: 0521865719. URL: https://nlp.stanford.edu/IR-book/html/htmledition/irbook.html.

[64] Christopher Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999. ISBN: 9780262133609.

[65] Viera Maslej-Krešňáková, Martin Sarnovský, Peter Butka, and Kristína Machová. "Comparison of deep learning models and various text pre-processing techniques for the toxic comments classification." In: *Applied Sciences (Switzerland)* 10.23 (2020), pp. 1–26. ISSN: 20763417. DOI: 10.3390/app10238631.

[66] Dirk Merkel. *Docker: lightweight Linux containers for consistent development and deployment*. 2014. URL: https://dl.acm.org/doi/10.5555/2600239.2600241.

[67] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. "Advances in pre-training distributed word representations." In: *LREC 2018 - 11th International Conference on Language Resources and Evaluation*. 2019, pp. 52–55. ISBN: 9791095546009. arXiv: 1712.09405. URL: https://commoncrawl.org/2017/06.

[68] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Machine learning*. 2012. ISBN: 978-0-262-01825-.

[69] David Nadeau and Satoshi Sekine. "A survey of named entity recognition and classification." In: (2007). DOI: 10.1075/li.30.1.03nad.

[70] Yoshihiko Ozaki, Yuki Tanigaki, Shuhei Watanabe, and Masaki Onishi. "Multiobjective tree-structured parzen estimator for computationally expensive optimization problems." In: *GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, 2020, pp. 533–541. ISBN: 9781450371285. DOI: 10.1145/3377930.3389817.

[71] Santisudha Panigrahi, Anuja Nanda, and Tripti Swarnkar. "A Survey on Transfer Learning." In: *Smart Innovation, Systems and Technologies* 194 (2021), pp. 781–789. ISSN: 21903026. DOI: 10.1007/978-981-15-5971-6_83.

[72] Drew Perkins. *Tagging Genes and Proteins with BioBERT | by Drew Perkins | Towards Data Science*. 2020. URL: https://towardsdatascience.com/tagging-genes-and-proteins-with-biobert-c7b04fc6eb4f (visited on 06/22/2022).

[73] Georgios Petasis, Alessandro Cucchiarelli, Paola Velardi, Georgios Paliouras, Vangelis Karkaletsis, and Constantine D Spyropoulos. "Automatic adaptation of proper noun dictionaries through cooperation of machine learning and probabilistic methods." In: *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*. 2000, pp. 128–135. DOI: 10.1145/345508.345563.

[74] *Plot confusion matrix sklearn with multiple labels - Stack Overflow*. URL: https://stackoverflow.com/questions/39033880/plot-confusion-matrix-sklearn-with-multiple-labels (visited on 10/13/2022).

[75] Pritesh Prakesh. *An Explanatory Guide to BERT Tokenizer - Analytics Vidhya*. 2021. URL: https://www.analyticsvidhya.com/blog/2021/09/an-explanatory-guide-to-bert-tokenizer/ (visited on 10/23/2022).

[76] Python Software Foundation. *PyPI · The Python Package Index*. URL: https://pypi.org/ (visited on 08/17/2022).

[77] Sebastian Ruder. *A Review of the Neural History of Natural Language Processing*. https://ruder.io/a-review-of-the-recent-history-of-nlp/index.html. 2018. (Visited on 07/14/2022).

[78] Sebastian Ruder and John G Breslin. "Neural Transfer Learning for Natural Language Processing." PhD thesis. 2019. URL: https://ruder.io/thesis/.

[79] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. "Transfer Learning in Natural Language Processing." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*. 2019, pp. 15–18. URL: https://ruder.io/state-of-transfer-learning-in-nlp/.

[80] Erik F. Tjong Kim Sang and Fien De Meulder. “Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition.” In: (2003). DOI: 10.48550/ARXIV.CS/0306050.

[81] Anne Schiller, Simone Teufel, Christine Stöckert, and Christine Thielen. “Guidelines für das Tagging deutscher Textcorpora mit STTS.” In: (1999). URL: https://telemaco.clarin-d.uni-saarland.de/hub/resource/304/.

[82] Jeff Schneider. *Cross Validation*. https://tinyurl.com/2p8x6ryk. 1997. (Visited on 07/14/2022).

[83] Wei Shi and Vera Demberg. “Next sentence prediction helps implicit discourse relation classification within and across domains.” In: *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*. 2019, pp. 5790–5796. ISBN: 9781950737901. DOI: 10.18653/v1/d19-1586. URL: https://github.com/google-research/.

[84] Singularity. *Singularity and Docker — Singularity container 2.6 documentation*. https://docs.sylabs.io/guides/2.6/user-guide/singularity_and_docker.html. 2018. (Visited on 08/17/2022).

[85] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. “A Survey of Optimization Methods from a Machine Learning Perspective.” In: *IEEE Transactions on Cybernetics* 50.8 (2020), pp. 3668–3681. ISSN: 21682275. DOI: 10.1109/TCYB.2019.2950779. arXiv: 1906.06821.

[86] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to sequence learning with neural networks.” In: *Advances in Neural Information Processing Systems* 4.January (2014), pp. 3104–3112. ISSN: 10495258. arXiv: 1409.3215.

[87] The Ray Team. *A Guide to Population Based Training — Ray 2.0.0*. 2022. URL: https://docs.ray.io/en/latest/tune/tutorials/tune-advanced-tutorial.html (visited on 08/24/2022).

[88] The Ray Team. *Ray Tune FAQ — Ray 1.13.0*. https://docs.ray.io/en/latest/tune/faq.html#which-search-algorithm-scheduler-should-i-choose. 2022. (Visited on 08/04/2022).

[89] Amirsina Torfi, Rouzbeh A. Shirvani, Yaser Keneshloo, Nader Tavaf, and Edward A. Fox. “Natural Language Processing Advancements By Deep Learning: A Survey.” In: (2020), pp. 1–23. arXiv: 2003.01200.

[90] Markus Unnewehr. *Arztbrief - Die Kommunikation optimieren*. https://www.aerzteblatt.de/archiv/145890/Arztbrief-Die-Kommunikation-optimieren. 2013. (Visited on 05/30/2022).

[91] Yuli Vasiliev. *Natural Language Processing with Python and SpaCy: A Practical Introduction*. 2020. ISBN: 9781718500525.

[92] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In: *Advances in Neural Information Processing Systems*. Vol. 2017-Decem. 2017, pp. 5999–6009. arXiv: 1706.03762.

[93] Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. "Automated concatenation of embeddings for structured prediction." In: *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*. 2021, pp. 2643–2660. ISBN: 9781954085527. DOI: 10.18653/v1/2021.acl-long.206. arXiv: 2010.05006.

[94] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing." In: (2020). DOI: 10.48550/arXiv.1910.03771. arXiv: arXiv:1910.03771. URL: https://github.com/huggingface/transformers.

[95] Yun Xu and Royston Goodacre. "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning." In: *Journal of Analysis and Testing* 2.3 (2018), pp. 249–262. ISSN: 25094696. DOI: 10.1007/S41664-018-0068-2.

[96] Vikas Yadav and Steven Bethard. "A Survey on Recent Advances in Named Entity Recognition from Deep Learning models." In: (2019). arXiv: 1910.11470.

[97] Andy B. Yoo, Morris A. Jette, and Mark Grondona. "SLURM: Simple Linux Utility for Resource Management." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 2862 (2003), pp. 44–60. ISSN: 16113349. DOI: 10.1007/10968987_3. URL: https://slurm.schedmd.com/slurm.html.

[98] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. "Dive into Deep Learning." In: *Journal of the American College of Radiology* 17.5 (2021), pp. 637–638. ISSN: 1558349X. arXiv: 2106.11342.

[99] Qi Zhu, Yuxian Gu, Lingxiao Luo, Bing Li, Cheng Li, Wei Peng, Minlie Huang, and Xiaoyan Zhu. "When does Further Pre-training MLM Help? An Empirical Study on Task-Oriented Dialog Pre-training." In: 2021, pp. 54–61. DOI: 10.18653/v1/2021.insights-1.9.

[100] *smanjil/German-MedBERT · Hugging Face*. 2021. URL: https://huggingface.co/smanjil/German-MedBERT (visited on 09/23/2022).