Hochschule Rhein-Waal
Rhine-Waal University of Applied Sciences
Faculty of Communication and Environment
Prof. Dr.-Ing. Rolf Becker
Dr. Lukas Gilz

Development of a Language model for the medical Domain

Master Thesis

by
Manjil Shrestha

Hochschule Rhein-Waal
Rhine-Waal University of Applied Sciences
Faculty of Communication and Environment
Prof. Dr.-Ing. Rolf Becker
Dr. Lukas Gilz

Development of a Language model for the medical Domain

A Thesis submitted in
Partial Fulfillment of the
Requirements of the Degree of

Master of Science
in
Information Engineering and Computer Science

by
Manjil Shrestha
Dietrichstr. 77
53175 Bonn

Matriculation Number:
24925

Submission Date:
03.09.2020

Abstract

Language models are widely used as a representation of written language in various machine learning tasks, with the most commonly used model being Bidirectional Encoder Representations from Transformers (BERT). It was shown that the prediction quality strongly benefits from language model pretraining on domain-specific data. The publicly available models, though are always trained on Wikipedia, news or legal data, thereby missing the domain specific knowledge about medical terms. In this thesis, we will train a BERT language model on medical data and compare performance with domain-unspecific language models. The dataset used for this purpose is the Non-technical Summaries - International Statistical Classification of Diseases (NTS-ICD) task of classification of animal experiment descriptions into International Statistical Classification of Diseases (ICD) categories.

Keywords: Multi-label Classification, ICD-10 Codes, Fine-tuning, BERT, Transfer Learning, NTS classification

Contents

List of Abbreviations

3R Replacement, Reduction, Refinement. 25, 27, 28

AI Artificial Intelligence. 14, 37, 100

BCE Binary Cross Entropy. xi, 82, 84, 85

BERT Bidirectional Encoder Representations from Transformers. iii, xii, 4–6, 9–18, 35–37, 39, 46, 52, 59, 66–70, 72–78, 80, 82–100, 107

BFR Federal Institute for Risk Assessment. 19

Bi-LSTM Bidirectional Long Short-Term Memory. 1, 3, 4

BoW Bag of words. 46, 47, 49, 50

CBOW Continuous Bag of Words. 82

CLS Text Classification. 15, 17

CLSTM Codes Attentive LSTM. 14

CNN Convolutional Neural Network. 14, 41, 44

CNS Central Nervous System. 23

DEP Dependency Parsing. 17

DF Document Frequency. 50

DL Deep Learning. 6

ELMo Embeddings from Language Models. 6, 8

F1 F1-score. 60, 61, 63–65, 67, 93–95

FN False Negative. 60–62, 64, 93–95

FP False Positive. 60–62, 64, 87, 93–95

GCTR German Clinical Trials Register. 14, 15

GM German Modification 2016 Version. 19, 29

ICD International Statistical Classification of Diseases. iii, 14, 19–23, 25, 28–32, 66, 67, 86, 87, 93–95, 98–100

IDF Inverse Document Frequency. 50, 51

IR Information Retrieval. 1

LM Language Model. 7, 8, 19, 46, 51, 52, 100

LSI Latent Semantic Indexing. 3

LSTM Long Short Term Memory. 1, 7, 8, 41, 45, 46

ML Machine Learning. 1–3, 6, 14, 19, 37–41, 58, 93

MLM Masked Language Model. 9, 12, 13, 73, 74, 79, 80, 98, 99, 107

NER Named Entity Recognition. 15–17, 99

NLI Natural Language Inference. 15, 17

NLP Natural Language Processing. 1–3, 6–8, 13–16, 19, 39, 45–47, 51, 52, 54, 59, 100

NSP Next Sentence Prediction. 9, 10, 13

NTS Non-technical Summaries. iv, 14, 15, 19, 23, 25–29, 31–34, 69, 70, 79, 80, 94, 95

NTS-ICD Non-technical Summaries - International Statistical Classification of Diseases. iii, xi–xiii, 14, 15, 31, 34, 49, 59, 60, 64, 66–72, 74, 80, 82–87, 93–95, 98–100

OAR Open Agrar Repository. 19

OOV Out of Vocabulary. 10, 16

OVA One-vs-All. 82

P Precision. 60–65, 67, 93

PICO PICO Extraction. 17

QA Question Answering. 15, 16, 99

R Recall. 60, 61, 63–65, 67, 93

RE Relation Extraction. 15, 16

REL Relation Extraction. 17

RNN Recurrent Neural Network. 1, 41, 45, 52

SVC Support Vector Classification. 81

SVM Support Vector Machine. 14

TF Term Frequency. 50, 51

TF-IDF Term Frequency - Inverse Document Frequency. 3, 14, 46, 50, 51, 67

TN True Negative. 61

TP True Positive. 60–62, 64, 95

ULM-FiT Universal Language Model Fine-Tuning. 6–8, 39

List of Figures

List of Tables

List of Equations

Chapter 1

Introduction

Language is one of the most essential things which makes us human, intellect and able to plan and communicate, and hence think. With the help of language, we can properly think, reason, and better communicate with others to express ourselves more clearly. However, as there are lots of languages available, it results in knowing the natural language to achieve fluent interaction and to effectively express our intents and expectations. It also enables us to receive vast amounts of information we would have never acquired without it. These types of natural languages are involved in all sectors like science, math, sociology, legal, medical and more activities that we perform daily in our life. Processing of all those information manually by a human can be time-consuming as well as the hidden information behind it cannot remain persistent. So, in the process of achieving persistency, these pieces of information are digitized and stored in computers which resulted in the evolution of *Natural Language Processing (NLP)* to help extract hidden information.

With the rise of *NLP*, it is now possible to perform interactions between computers and natural languages by applying machine learning algorithms to understand text and speech. It also served as the intersection of artificial intelligence and linguistics. The purpose of *NLP* is mostly to extract meaning from texts like *part-of-speech* tagging which distinguishes a word to be either a noun, a verb, or an adjective. Brants (2003) mentions various *NLP* techniques, including *stemming*, *part-of-speech tagging*, *chunking*, *word sense disambiguation*, and others that have been used to perform several *Information Retrieval (IR)* tasks like document classification, text summarization, speech recognition, spam detection, machine translation, named entity recognition. *NLP* is more concerned in capturing the syntax and semantic representations of the text and has been a strong foundation on building these *IR* applications.

While *NLP* is more related to perform linguistic tasks and attempting to extract contextual information using approaches like *Long Short Term Memory (LSTM), Bidirectional Long Short-Term Memory (Bi-LSTM), Recurrent Neural Network (RNN), attention, and transformers* which we will discuss in chapter 4, *Machine Learning (ML)* provides with

the statistical knowledge to perform various tasks such as classification and clustering. *ML* helps the machine to learn the representations developed by various *NLP* techniques. *ML* as said by Roesch et al. (2018), powers diverse services in industry including search, translation, recommendation systems, and security. *ML* allows computers to discover the insights from the sources to perform the underlying tasks without being explicitly programmed to do so. All it needs is the dataset to learn the model from and the task to perform like *document classification*, *machine translation*, *recommendation*, and many more. The performance of the resulting model might vary depending on the volume of the dataset it used during training the model. As mentioned by Fortuny, Martens, and provost (2013), larger datasets will often lead to richer models with the better predictive ability with also the chance of over-fitting. But, with proper fine-graining and feature selection from larger datasets, the performance of the predictive analytics gets better than the one with fewer datasets.

There are a huge number of datasets available to perform the tasks mentioned above, but the dataset volume may vary in size which can lead the model to not perform so well. There often occurs a case when there is a low volume of the data on which we have to perform a predictive task. And, as argued in Fortuny, Martens, and provost (2013), small datasets do not perform very well in certain *ML* algorithms. This is one of the reasons for the development of the *transfer learning* concept. Transfer learning as mentioned by Rosenstein, Marx, and Kaelbling (2005) involves two interrelated learning problems to use knowledge about one set of tasks to improve performance on a related task. Do and Ng (2006) also mentions about *transfer learning* while attempting to solve a text classification problem by *meta-learning* a new *learning algorithm* which automatically chooses a appropriate parameter function to apply to a new classification problem. This is an important approach to utilize when we have a relatively small number of datasets to train and it helps to get better performance by utilizing the word-embeddings and pre-trained word vectors from a pre-trained model in a similar task. And, this pre-trained model in case of a text classification problem would be a pre-trained language model that already has a set of pre-trained word-embeddings which helps to represent the text to an understandable form for the machine.

To better classify and represent natural languages, several language models have been built to retrieve inferences like named entity recognition, document classification, and many more. But, these language models are mostly built for the English language. Proper language models have always proven to be useful to get the word meanings and context from the sentence. Although there are sufficient language models developed for this kind of work, very few exist which are domain-specific. With this scenario in mind, this paper tries to achieve the functionalities of a proper language model by feeding it with some domain-specific texts. As a domain, we are concerned with the behavior of a language

model with medical texts in german language. Medical texts could contain either the symptoms, diagnosis, history of the patient, the medications, therapies, procedures taken, or the collection of all. And, referring to all those medical texts, the language model would extract the features to build a word context to better represent it with a diagnosis code. Keeping this in mind, these language models can also be used to build a classification model that can better represent the medical texts for proper inferences.

This type of problem is treated as a multi-label classification problem, as a patient could be associated with either a single disease or multiple diseases. This paper works on relating the medical texts with accurate diagnosis codes associated with it.

1.1 Motivation

Text classification has been a widely used application in the domain of *NLP*. The main purpose of it is to link the underlying text document to its related group whether it be *sentiment analysis*, *document classification*, *spam detection*, or *named entity recognition*. Also, it has evolved in such a way that each document can now be mapped to either one or multiple groups and over the past few decades, much research has been done for text classification problems.

Do and Ng (2006) utilizes multiclass text classification to map given training set of documents to one of disjoint classes. In their paper, they consider the task of automatically learning a parameter function for text classification by proposing a "meta-learn" approach. The "meta-learning" approach they propose leverages data from a variety of related classification tasks to obtain a good classifier for new tasks. And this, as they say is an instance of *transfer learning* which outperformed other approaches which used human-designed parameter functions based on *Naive Bayes* and *Term Frequency - Inverse Document Frequency (TF-IDF)*.

Another text classification problem was handled by Dalal and Zaveri (2011), which involved assigning a text document to a set of pre-defined classes automatically, using a *ML* technique and done on the basis of significant words or features extracted from the text document. Their paper explains about the general pre-processing techniques used for texts such as eliminating stop-words or stemming, and some major issues during their work like delaing with unstructured text, handling large number of attributes. To handle large number of attributes, the authors experimented with *TF-IDF*, *Latent Semantic Indexing (LSI)*, and *multi-word*. Their paper also discusses about the need of *text classifiers* for languages other than english and argues about which supervised machine learning technique would be better for classification of text documents. P. Zhou et al. (2016) proposes *Bi-LSTM* Networks with Two-Dimensional Max Pooling (BLSTM-2DPooling) to

capture features on both the time-step dimension and the feature vector dimension where it first utilizes *Bi-LSTM* to transform the text into vectors and then 2D max pooling operation to obtain a fixed-length vector. Besides that, they also apply 2D convolution to capture more meaningful features to represent the input text. The authors here make use of *Bi-LSTM* to grab the future context as well as the past context. The authors report that *Bi-LSTM*-2DCNN performed better than *Bi-LSTM*-2DPooling.

Adhikari et al. (2019) utilizes *BERT* for document classification where it minimizes the computational expense from $BERT_{\text{large}}$ to $BERT_{\text{base}}$ by distilling knowledge using 30X fewer parameters. They *fine-tune* all parameters of *BERT* to achieve state-of-the-art results for document classification. They report that choosing a batch size of 2e-5, and maximum sequence length of 512 tokens yields better performance on the validation sets of all datasets. Also, for multi-label tasks after learning the representations for $BERT_{\text{large}}$, they used a batch size of 128 and 64 for single-label tasks.

With all these approaches already available for *text classification*, *document classification* in english language and general domain, this paper works on utilizing *German BERT*[1] model and *fine-tune* it with medical texts to verify the effects of *transfer learning* and *fine-tuning* on domain-specific data.

1.2 Problem Statement

Language models for a specific domain are highly required as to make it more reliable to the conclusions it can draw from the domain data. As most of the words from medical texts can contain words with specific medical terminology that is difficult for a general language model to capture. Hence, the result with general language models would not be so reliable to make further decisions using it. There can be ambiguity between some words which results in the language model to capture incorrect context.

In this case, if the existing *German BERT* language model is *fine-tuned* using medical texts, the model could capture the representations for medical terms and better learn from the medical context. So, if we consider the medical terms like **Krebs**, **neurogenerativen**, **Hypertonie**, **Diabetes** which are specific and special medical terms related to disease or health situation it will be difficult to capture it by the general *German BERT*.

Therapie gegen **Stammzellen** beim **Darmkrebs** **Darmkrebs** ist eine der häufigsten **Krebserkrankungen** weltweit und endet häufig **tödlich**.

Referring to the two sentences above, the model learns the representations related to the

[1] Deepset(German BERT) https://deepset.ai/german-bert

words **Krebs**, **Darmkrebs** and build contextual relation between them. The language model thus built using this domain dataset will prove useful to better understand the relativity of the meaning and context of the sentences to draw better diagnosis code suggestions related to the disease "Cancer" or any other diseases.

1.3 Project Objective

Although, there are various language models available which are trained with general texts with no specific domain, we intend to *fine-tune* the existing *German BERT* language model with texts from the medical domain in german language. Through this paper, we want to verify that *fine-tuning* a language model with domain-dependent text can better represent the underlying terminologies and inference them in an efficient manner.

The current language domain being considered is the medical field in the German language. As German *BERT* has been trained using Wikipedia, open legal data dump, and news articles, this model has some shortcomings in performance since it does not contain any representations for medical terms. We will, therefore, fine-tune this German *BERT* language model with articles from the medical domain to ensure more robust representations in the target domain and hence improve its performance on the downstream task.

Given a medical text, the underlying language model should be able to identify medical entities, diseases, symptoms or any therapies. This will be verified by observing the performance of the language model on the target tasks. And then, it should be able to perform the downstream task which is *multi-label text classification* as per our thesis scope. Thus, this language model is most suitable for building systems for medical text classification and get its underlying benefit such as predicting the diagnosis described the patient.

In this paper, we try to answer the following research questions and investigate it:

1. Does fine-tuning improve the quality of language models on the target domain?
2. Does the fine-tuned model improve model performance for downstream tasks?
3. Does improvement in language model also imply improvement in downstream tasks?

Chapter 2

Related Works

In this chapter we will discuss about the works that are related to the transfer learning and using context vectors. We will at the end look into *BERT* language model which has been a huge development using *Deep Learning (DL)* approaches. *BERT* release has created a huge impact on this type of research and several models has been developed to achieve results on different domains like science (Beltagy, Lo, and cohan 2019) and general languages. We also review different *NLP* research that were available before *BERT* was developed and how *DL* approaches evolved over classical *ML* approaches.

2.1 Transfer Learning

In this section, we will be discussing about *transfer learning*, its uses, importance and applications in *DL*. We will also look at the *word-embeddings* and how it inspired the development of various *transfer learning* approaches like *Embeddings from Language Models (ELMo)* and *Universal Language Model Fine-Tuning (ULM-FiT)*.

Whenever we are training a neural network model or a *DL* model from scratch, we will be requiring a lot of data for the model to learn well. But in most cases, we will not always have access to that volume of data. This limitation of data volume can lead the model to not learn the representations well and the underlying prediction result from the model will not seem so promising. So, this is the situation where *transfer learning* becomes the savior. *Transfer learning* builds a new language model by utilizing the *word-embeddings* or the *learnings* from an existing model which is also known as a *pre-trained* model. Not only *transfer learning* can be used when there is less volume of data, but also when there is already a model trained with a similar task on a high volume of data.

Transfer learning relaxes the hypothesis that the training data must be independent and identically distributed with the test data, which motivates us to use transfer learning to solve the problem of insufficient training data (Tan et al. 2018). It is a technique to transfer the knowledge from the source domain to the target domain which also makes it an important tool in machine learning to solve the problems of insufficient training data.

Transfer learning mostly makes it helpful when the training data is insufficient in some special domains and collecting more data is quiet inefficient and expensive.

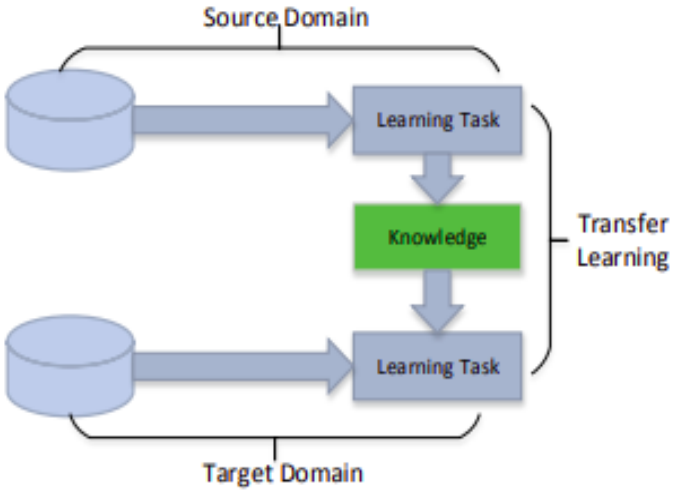


Figure 2.1: Learning process of transfer learning. (Taken from Tan et al. 2018)

As seen in Figure 2.1, *transfer learning* transfers the knowledge from the *source domain* to the *target domain*, where in most cases, the size of datasets from *source domain* is much larger than the size of datasets from *target domain*. At this stage, the datasets on the *source domain* are already trained with a larger dataset resulting in a *pre-trained* model. The idea of *transfer learning* is to only use the pre-trained model's weights to extract features but not to update the weights of the pre-trained model during training with target data for the target task.

2.1.1 ULM-FiT

ULM-FiT is a research on an effective transfer learning method that can be effective for any task in *NLP* as presented by Howard and Ruder (2018) and also introduces techniques that can be effective for fine-tuning a language model. The authors also proposed this method to pre-train a *Language Model (LM)* on a large general-domain corpus and fine-tune it on the target task using novel techniques.

During their experiment, they used *AWD-LSTM*, a regular *LSTM* with no attention, shortcut connections, or other sophisticated additions, and with various tuned dropout hyperparameters. They also point out different fine-tuning approaches like using *discriminative fine-tuning* to use different learning rates for each layer and *slanted triangular learning rates* where it first linearly increases the learning rate and then linearly decays it according to a certain update schedule.

Howard and Ruder (2018) proposes to gradually unfreeze the model starting from the last layer and fine-tune all unfrozen layer, and continue until all layers are unfrozen and fine-

tuned. The researchers of this paper claim that using discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing is beneficial. This paper also points out that the general-domain pretraining and novel fine-tuning techniques to prevent overfitting even with 100 labeled examples achieves state-of-the-art results also on small datasets.

ULM-FiT being a *pre-trained* language model on a large general-domain corpus which then *fine-tunes* it on the target task using novel-techniques like *discriminative fine-tuning, learning-rates*, seemed to be an effective approach of *fine-tuning* on the target task with a *pre-trained* model. Additionally, it is also a better approach that handled *transfer learning* methods by using the *pre-trained* word embeddings which were helpful to capture the long-term dependencies for the target task.

2.1.2 ELMo

ELMo is another transfer learning approach proposed by Peters et al. (2018) which was published almost together with *ULM-FiT*. This model is a new type of deep contextualized word representations that models both complex characteristics of word use and how to model polysemy. The author also claims that these representations can be easily added to existing models and significantly improve state of the art across several *NLP* problems.

ELMo uses vectors derived from a bi-directional *LSTM* that is trained with a coupled language model objective on a large text corpus. A bi-directional language model is a combination of the forward and backwards *LMs* that are trained separately. *ELMo* word representations are functions of the entire input sequence which are computed on top of two-layer bi-directional *LMs* with character convolutions as a linear function of the internal network states.

In *ELMo*, the syntatic information is better represented at lowest layers while semantic information is captured at higher layers where the bi-directional *LM* is able to disambiguate both the part of speech and word sense in the source sentence. The obtained word embeddings from *ELMo* are compatible with any target task and adding it to a model increases the sample efficiency. *ELMo* could achieve the state-of-the-art result at that time along six different *NLP* tasks.

ELMo, being the one which also captures the syntactic and semantic information of word utilizing the *deep bi-directional contextualized* mechanism, it can handle the limitation of Mikolov et al. (2013) and Howard and Ruder (2018) by also modeling *polysemy*. *Polysemy* refers to a word which has various meanings. The word **bank** in the two sentences *I got money from a bank* and *I sit on a bank* relates to two different meanings, where the first one would say *the place to save your money* and the second would be *the river bank*. This difference is handled by *ELMo* while *word2vec* would construct a relation between

money and *sit* seeing that both words are closer to *bank*.

2.2 Bidirectional Encoder Representations from Transformers

In this section, we will discuss about *BERT*, and its approaches on *pre-training* and *fine-tuning*. Besides it, we will see how *BERT* constructs its input representation to be used in the training layers. We will see a sample tokenization by *BERT* which uses its own *WordPieceTokenizer*, and also have a look on its masked token prediction.

BERT, which stands for *Bidirectional Encoder Representations from Transformers* is designed to train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. *BERT* uses a *Masked Language Model (MLM)* and *Next Sentence Prediction (NSP)* pre-training objective.

BERT uses two-phase training known as: pre-training and fine-tuning. The *MLM* of *BERT* randomly masks some of the tokens from the input sequence, and the objective is to predict the original vocabulary id of the masked word based only on its context. It follows several steps to replace "masked" words with actual [MASK] tokens (Devlin et al. 2019):

- replace i-th token with the [MASK] token 80% of the time.
- replace i-th token with a random token 10% of the time.
- replace i-th token with the unchanged i-th token 10% of the time.

And, *BERT* uses the fusion of both left-to-right and right-to-left model to demonstrate deep bidirectional model insted of individual left-to-right or right-to-left model. It also pre-trains for a binarized next sentence prediction task.

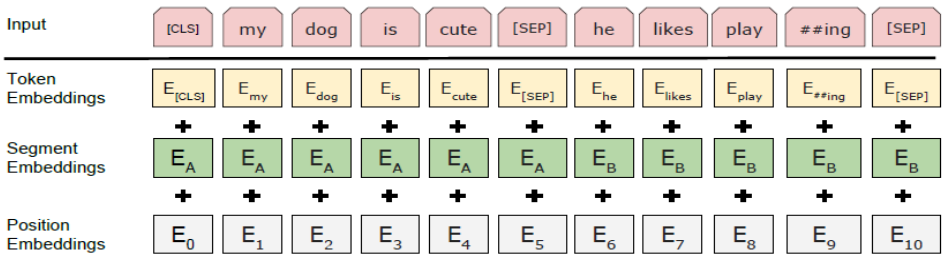


Figure 2.2: BERT input representations. (Taken from Devlin et al. 2019)

The Figure 2.2 shows the construction of input representation by *BERT*. *BERT*'s model architecture is a multi-layer bidirectional transformer encoder, where it also uses bidirectional self-attention. *BERT* uses three different type of emeddings to construct its input representation: *token embeddings*, *segment embeddings*, and *position embeddings*. We

will not step-wise see how each of these embeddings are created to form a input representation.

> From Figure 2.2, we have two sentences:
> **Sentence A**: My dog is cute.
> **Sentence B**: He likes playing.

As per the *BERT* input representation, the first step is tokenizing the input sentence and adding special tokens like [CLS] and [SEP] before passing it to the *Token Embeddings*. [CLS] token is the first token of every input sentence which in terms of the classification task is helpful to identify the start of the sentence, and [SEP] token is used to identify the end of the first sentence and start of the another in case of two sentences.

> Here, we see a sample tokenization by *BERT* WordPieceTokenizer (Devlin et al. 2019).
> **Sentence A**: [CLS], my, dog, is, cute, [SEP]
> **Sentence B**: [CLS], he, likes, play, ##ing, [SEP]

After the tokenization, we see that the word *playing* is separated into two pieces, which is done by the *WordPiece* tokenization method to create a balance between vocabulary size and *Out of Vocabulary (OOV)* words and allowing *BERT* to store 30,000 words in its vocabulary. While tokenizing, if a word is separated in multiple pieces then the pieces except the first word is prepended with ##. The tokenization from above resulted in a token length of size 6 for both *Sentence A* and *Sentence B*.

The *Token Embeddings* layer also makes use of another parameter in addition to token length known as *hidden size*, which in case of $BERT_{base}$ model is 768 and $BERT_{large}$ model is 1024. This results in our sentence of 6 input tokens to be converted into a matrix of shape (6, 768) and (6, 1024) for $BERT_{base}$ and $BERT_{large}$ models respectively.

Now, we will proceed to *Segment Embeddings*, and see how they are created to properly handle the sentences pair which is careful for classification tasks like *NSP*, and *Question Answering*. To train for *NSP* tasks, *BERT* creates a sentence pair by *collecting two neighboring sentences*, which is labelled as *IsNext* and by *collecting two random sentences*, which is labelled as *NotNext*.

> Here, we will see the *Sentence Embeddings* performed for the input sentences seen above.
> First, we concatenate both sentences A and B and create a single input by concatenating and tokenizing both which results as:
> [CLS], my, dog, is, cute, [SEP], he, likes, play, ##ing, [SEP]
>
> Now, as we see that both sentences are represented together as a sequence pair by use of [SEP] token, it is still not sufficient. So, we label those two sentences with 0 for the first sentence and 1 for the second sentence to uniquely identify the sentences, which results as:
> 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1

The *Segment Embeddings* created above for the sentence pair resulted in a token length of 11 tokens. And, hence as done in *Token Embeddings*, this will be converted into a matrix of shape (11, 768) and (11, 1024) for $BERT_{base}$ and $BERT_{large}$ models respectively.

The third or the last part of the input representation is the *Position Embeddings* which is only used to indicate the position of the token in the sequence. As *BERT* can handle input sequences up to the length of 512, the *Position Embeddings* would be a matrix of shape (512, 768) or (512, 1024).

And finally, all the three embeddings are combined to form a robust input representation which has better vocabulary from *WordPiece* representation, identifies the boundary of a sequence, and knows about the position of a token to better represent the context of the sequence by capturing the order of appearance of the token which helps predict missing tokens from the sequence.

During *pre-training*, *BERT* randomly masks the input tokens and is trained to predict the original token id of the masked token based on its context using the bidirectional language model.

Here, we will see a simple example of token masking and prediction:
Sentence = "My dog is cute. He likes playing."

First here, we mask the word *cute* only in the sentence *My dog is cute.* and let the model predict the mask token.

Masked Sentence = "My dog is [MASK]."

The result of the prediction here is:

Table 2.1: BERT mask prediction without context.

Token	Score
dead	17.6%
gone	6.9%
here	6.3%
fine	4.5%
mine	2.8%

Here, without providing any context, the first result of the prediction in Table 2.1 is *dead* which is sad as the model here only learnt *grief*. Now, let us make this a better place by providing the model with more context by combining both sentences as:

Masked Sentence = "My dog is [MASK]. He likes playing."

The result of the prediction here is:

Table 2.2: BERT mask prediction with context.

Token	Score
good	9.6%
cute	6.3%
nice	6.3%
fine	4.6%
great	4.3%

This result in Table 2.2 looks more happy and cheerful now. As we can see that the original token is already present in the second place, the first token is also a better representation of a dog who is playing, happy and cheerful.

It also trains it for next sequence prediction by feeding it with a sequence of sentence pairs from same document and random sentence pairs from different documents. *Fine-tuning* in *BERT* is faster compared to *pre-training*. It also accepts the *MLM* objective to train

it for the target task as compared to *pre-training*. To perform *fine-tuning* on any target task, we simply plug in the task specific input and outputs into *BERT* and *fine-tune* all the parameters end-to-end. $BERT_{base}$ is a model with 12 layers and $BERT_{large}$ is a model with 24 layers. Its *pre-training* is performed with a *BooksCorpus (800M words)* and *English Wikipedia (2,500M words)* and achieved state-of-the-art performance on eleven *NLP* tasks. For *fine-tuning*, Devlin et al. (2019) mentions that epochs of 2 to 4, learning rate of 2e-5, 3e-5 or 5e-5 and batch size of 16, 32, 64 was optimal to help achieve those scores.

2.2.1 Pre-training

The *pre-training* of *BERT* has been performed for 40 epochs over a 3.3 billion words of English corpus using the *English Wikipedia* dump and *BookCorpus* as training dataset (Devlin et al. 2019). *BERT* has two pre-training objectives known as *MLM* and *NSP* which we will discuss below.

Masked Language Model

BERT masks a certain number of tokens from the input sequence to train and see how well the underlying language model can predict the missing token. The part of tokens to be masked has been provided by Devlin et al. (2019) and has been discussed in Section 2.2. This works as a self-supervised training setup which we discussed in Section 4.1.3 in which the language model is tasked to predict only the masked tokens. And, the model further is evaluated on how good it was able to learn using the bi-directional representation and predict the masked tokens with minimal loss.

Next Sentence Prediction

NSP is another pre-training objective of *BERT* where it feeds in two sentences and tries to learn if the sentences are related or not. Hence, the motive of this training is to enhance the understanding of the relationship between sentences. The sentences are chosen in two different ways where:

1. sentence B is the next sentence of A.
2. sentence B is randomly chosen and therefore not related to A.

This is more like supervised learning where both cases are given their respective labels as *isNext* and *notNext* to perform the training.

2.2.2 Fine-tuning

Fine-tuning is an effective approach to apply *transfer learning* as discussed in Section 2.1 to neural networks by copying and training the first n layers. There are two distinct approaches for *fine-tuning* or to apply the *pre-trained* language representations for

downstream tasks: *feature-based* and *fine-tuning* as mentioned in Devlin et al. (2019). In *feature-based* approach, only the embeddings are used while the rest of the layers are trained, and in *fine-tuning* which also means the *frozen layer* approach, we can freeze all the layers leaving only one top layer to be trained or depending on the application. The reason for freezing the lower layers and only training the top layers is that, the lower layers comprises of the syntactic and semantic representation which can later be useful while training for *target tasks*, hence saving time and resource.

2.3 NLP research in NTS

The way of doing work in medical domain has been changing due to the development on *NLP* and *Artificial Intelligence (AI)*. *NLP* helps to give better code suggestions by taking in the patient history as text and learning representations. The methods to overcome these things are improving as we can see some number of researches being done in biomedical domain. In this section, we will discuss about the previous approaches done to classify *NTS-ICD10* documents. *NTS-ICD10* is a dataset containing the text with clinical experiments conducted in animals which are linked to their respective *ICD10* codes. We will look into *NTS-ICD10* dataset and the *ICD10* codes in more detail in chapter 3.

2.3.1 Classification of NTS Documents

Amin et al. (2019) and Sänger et al. (2019) published their work on classification of medical documents for german *NTS-ICD10* datasets using some traditional *ML* techniques like *TF-IDF baselines wih linear Support Vector Machine (SVM), Convolutional Neural Network (CNN), Codes Attentive LSTM (CLSTM)* and also comparing it with *BERT*.

Amin et al. (2019) applied various architectures like pre-processing, translating, including *pre-trained* word-embeddings in multi-label classification settings and demonstrate the effectiveness of transfer learning with pre-trained language representation model *BERT* and its recent variant *BioBERT*. They also translated the task documents from *German* to *English*, and then compare the performance between those two datasets. The classification was performed on predicting the *ICD-10* codes for the *NTS* documents. The authors report result of 73.02% *F1-micro* score in classification *ICD-10* code using *BioBERT*.

Another comparative study of *NTS-ICD10* code classifiers was done by Sänger et al. (2019). This study also approached the task as multi-label classification problem and leverages the multi-lingual version of the *BERT* text encoding model to represent the summaries. In addition to the *NTS-ICD10* dataset, they also made use of extra training data from the *German Clinical Trials Register (GCTR)* and ensemble several model instances to improve the overall performance. They compare their model with

five baseline systems including a dictionary matching approach and single-label SVM and *BERT* classification models, and their model achieved a *F1* score of 80% in the final evaluation.

Amin et al. (2019) also reports the *F1-micro* score for development set of 76.68% with multi-lingual *BERT* on german *NTS-ICD10* dataset and 82.90% with *BioBERT* on english dataset which they translated from german *NTS-ICD10* dataset and reported as their best score. Sänger et al. (2019) reports the *F1* score for development set of 74.7% with *BERT multi-label avg* on *NTS* dataset and 81% on combined *NTS* and *GCTR* dataset.

Amin et al. (2019) reports that translated texts (English) improved the score by an average of 4.07% which could be due to the fact that there is much more English texts than other languages. *BERT* performed better than other models both in *German* and *English* with an average score of 6% points higher, whereas Sänger et al. (2019) reports that with *BERT* model, the performance increases by 10.4%.

2.4 Domain Specific BERT Models

After the development of *BERT*, several approaches were tested using *BERT* to see the domain adaptation capabilities by fine-tuning the language model with domain specific datasets which led to the discovery of medical domain pre-trained *BERT* language models like *BioBERT*, *SciBERT*, and *ClinicalBERT*. In this section we will see the approaches used by the language models mentioned above, and how they achieved the state-of-the-art performance in various *NLP* tasks like *Named Entity Recognition (NER)*, *Relation Extraction (RE)*, *Question Answering (QA)*, *Text Classification (CLS)*, and *Natural Language Inference (NLI)* using domain specific datasets.

2.4.1 BioBERT

BioBERT (Lee et al. 2019) is the first domain-specific *BERT* based model optimized for biomedical field. *BioBERT* uses 4.5B words from PubMed Abstracts and 13.5B words from PMC Full-text articles from biomedical domain also combined wih 2.5B words from English Wikipedia and 0.8B words from BooksCorpus from general domain. The pre-training for this model was reported to last for 23 days on eight NVIDIA V100 GPUs.

Combining the words from general domain with biomedical domain is due to the reason that *BioBERT* is initialized with $BERT_{base}$ which means that the language model is built on top of the learned weights from English Wikipedia and BooksCorpus. The authors also found that when using both PubMed and PMC corpora, 200K and 270K pre-training steps were optimal for PubMed and PMC respectively which was released as *BioBERT v1.0*. Later, they pre-trained only using PubMed for 1M steps and released

BioBERT v1.1 improving the state-of-the-art performance over *BioBERT v1.0*.

BioBERT tested its performance on 14 task-specific datasets from biomedical domain specially from *NER*, *RE*, and *QA*. *NER* itself contains nine datasets belonging to four different groups namely *disease*, *drug/chem*, *gene/protein*, and *species*, while *RE* contains three datasets from two relation groups like *gene-disease*, and *protein-chemical* (Lee et al. 2019). The remaining tasks *QA* had three different *BioASQ* factoid datasets provided by BioASQ organizers. This model achieves the state-of-the-art result in six out of nine datasets from *NER* and *BioBERT v1.1* outperformed the state-of-the-art models by 0.62 in terms of micro averaged F1 score. This *BioBERT* model also achieves state-of-the-art result in 1 out of 3 *RE* dataset. It also achieved new state-of-the-art performance in all 3 *QA* datasets. Figure 2.3 shows the *pre-training* datasets and architecture of *BioBERT* and later *fine-tuning* it with some *task-specific* datasets to achieve state-of-the-art results mentioned above.

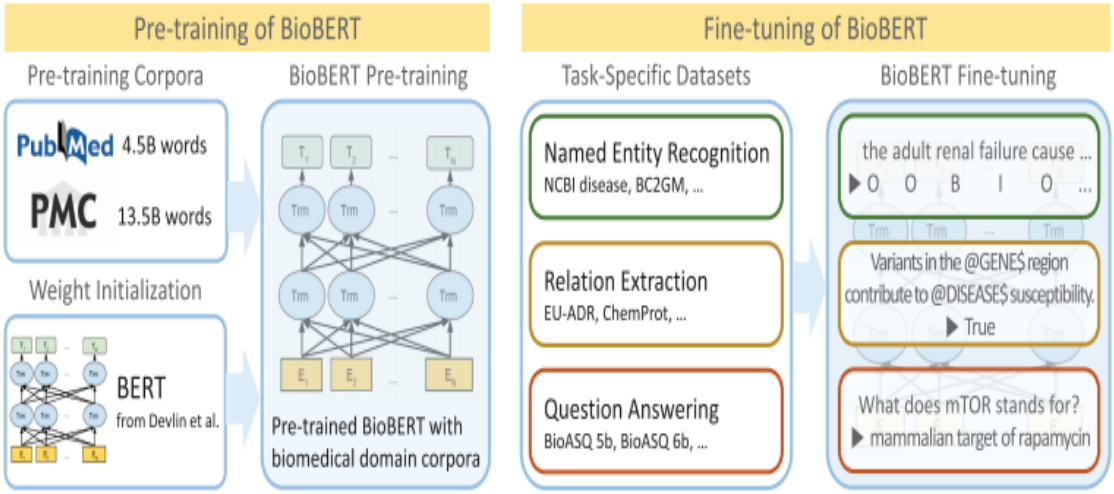


Figure 2.3: Overview of the pre-training and fine-tuning of BioBERT. (Taken from Lee et al. 2019)

Lee et al. (2019) relied on the *WordPiece* tokenizer of *BERT* to handle *OOV* words by separating them into sub-words and did not create their own vocabulary so that *BioBERT* and *BERT* would be compatible and easier to interchangeably use existing models. For *fine-tuning*, the author used a batch size of 10, 16, 32 or 64, and a learning rate of 5e-5, 3e-5 or 1e-5 which are also the values recommended by *BERT*.

2.4.2 SciBERT

SciBERT (Beltagy, Lo, and cohan 2019) is a pre-trained language model based on *BERT* to address the lack of high-quality, large-scale labeled scientific data which leverages unsupervised pre-training on a large multi-domain corpus of scientific publications to improve performance on downstream scientific *NLP* tasks.

SciBERT constructs its own *SCIVOCAB*, a new *WordPiece* vocabulary on its scientific corpus using *SentencePiece*[1] library. *SciBERT* was pre-trained using 1.14M papers from *Semantic Scholar* which consists of 18% papers from computer science domain and 82% from the broad biomedical domain. The size of the corpus is 3.17B tokens and is trained using a single TPU v3 with 8 cores. Training the *SCIVOCAB* on this corpus took 1 week where in the first 5 days it was trained with a maximum sentence length of 128 tokens and in the last 2 days it was trained with a maximum sentence length of 512 tokens.

SciBERT tested its performance on 12 task-specifc datasets from *NER*, *PICO Extraction (PICO)*, *CLS*, *Relation Extraction (REL)*, and *Dependency Parsing (DEP)* which belonged to biomedical, science and multi domain. This model outperforms $BERT_{base}$ on biomedical tasks. It also achieves state-of-the-art results in 8 out of 12 tasks, and outperforms *BioBERT* on two tasks, one is *NER*, and the other is *REL*. Beltagy, Lo, and cohan (2019) also reports that using *SCIVOCAB* instead of $BERT_{base}$ *vocab* resulted in a *F1* score of 60% on average across all datasets. And, the author reports that the optimal setting for *fine-tuning* was 2 or 4 epochs, and a learning rate of 2e-5.

2.4.3 ClinicalBERT

ClinicalBERT (Alsentzer et al. 2019) is the first released pre-trained *BERT* language model in clinical domain. *BioBERT* was present as a medical domain text but mostly focused on biomedical text, so, *ClinicalBERT* known as the specialized clinical *BERT* model overcomes the missing need of handling clinical narratives (e.g., physician notes) which have known differences in linguistic characteristics from both general text and non-clinical biomedical text.

Alsentzer et al. (2019) uses data from the approximately 2 million notes in the *MIMIC-III v1.4* database, and trains two variety of *BERT* on *MIMIC* notes: *Clinical BERT*, which sees text from all note types, and *Discharge Summary BERT*, which uses only discharge summaries. Besides training on *MIMIC* notes, *ClinicalBERT* also trains two *BERT* models on clinical text: one is *Clinical BERT*, initialized from $BERT_{base}$, and another is *Clinical BioBERT*, initialized from *BioBERT*.

Training *BERT* on *MIMIC* notes took roughly 17-18 days using a single *GeForce GTX TITAN X 12 GB GPU* excluding the time required for pre-processing or downloading *MIMIC* dataset. *ClinicalBERT* and *Clinical BioBERT* tested its performance on one *MedNLI NLI* task, and four *idb2 NER* tasks.

Out of all the five tasks, Alsentzer et al. (2019) reports that on three tasks *MedNLI, idb2*

[1] SentencePiece https://github.com/google/sentencepiece

2010, and idb2 2012, clinically fine-tuned *BioBERT* shows improvements over *BioBERT* or general *BERT*, and *ClinicalBERT* achieves a new state-of-the-art on *MedNLI* scoring a performance of 82.7% over 73.5%. The author also mentions that it is useful to use domain-specific contextual embeddings for *ClinicalNLP* tasks.

Chapter 3

Dataset Description

For every *ML* or *NLP* work, the main requirement for the implementation of the algorithm is the *dataset* that helps the underlying model to function well. As mentioned in chapter 1, we are interested in *fine-tuning* the existing *LM*, and *evaluate* the performance which results in utilization of two datasets: one to *fine-tune* the *LM* and the other to *evaluate* the model. So, in this Chapter we will be discussing about the dataset used for this thesis work along with the origination and type of dataset. We will also have a look on certain insights on the *evaluation* dataset like *code frequency per document*, *label distribution* and similar in Section 3.3. But, to get a better understanding of the labels shown in Section 3.3, we will also get some insights about the labels in Section 3.1 which has a brief description about *ICD10* Chapters and groups. In Section 3.2, we will also get a deeper look inside one document from the dataset. And, finally in Section 3.4, we will also get a brief overview on the dataset we used for *fine-tuning*.

The *evaluation* dataset used in this paper is *NTS* of animal experiments carried out in Germany indexed with *ICD-10* Chapters and groups belonging to *German Modification 2016 Version (GM)*. This dataset contains the information about various experiments performed on various animals having some diseases with some artificially created to achieve better approach on identifying the treatment solutions. The *NTS* is a clear and detailed document that provides a description of the animal experiments and its findings in a way that is both easy to read and understand requiring very less pre-processing for further evaluation steps. This *evaluation* dataset is available in *Open Agrar Repository (OAR)* which is originally available at the *AnimalTestInfo* database published by *Federal Institute for Risk Assessment (BFR)*. [1] [2] [3]

Figure 3.1 shows the distribution of the *evaluation* dataset. The *evaluation* dataset contains a total of 8,386 documents (training and development sets), and 407 test documents, all in German. 7,544 documents from the total dataset are used as a training dataset, and

[1] Open Agrar Repository https://www.openagrar.de/receive/openagrar_mods_00046540?lang=en

[2] AnimalTestInfo database https://www.animaltestinfo.de/

[3] BFR https://www.bfr.bund.de/en/home.html

the remaining 842 documents are used as development dataset. Out of the toal documents, only 6,508 documents have been labelled with their respective Chapters and their groups from *ICD-10*, while the remaining 1,878 documents are not labelled and, out of the labelled datasets, 5854 belongs to the training set and 654 belongs to the development set. Also, this dataset consists of 233 unique labels belonging to the *ICD-10* Chapters and groups.

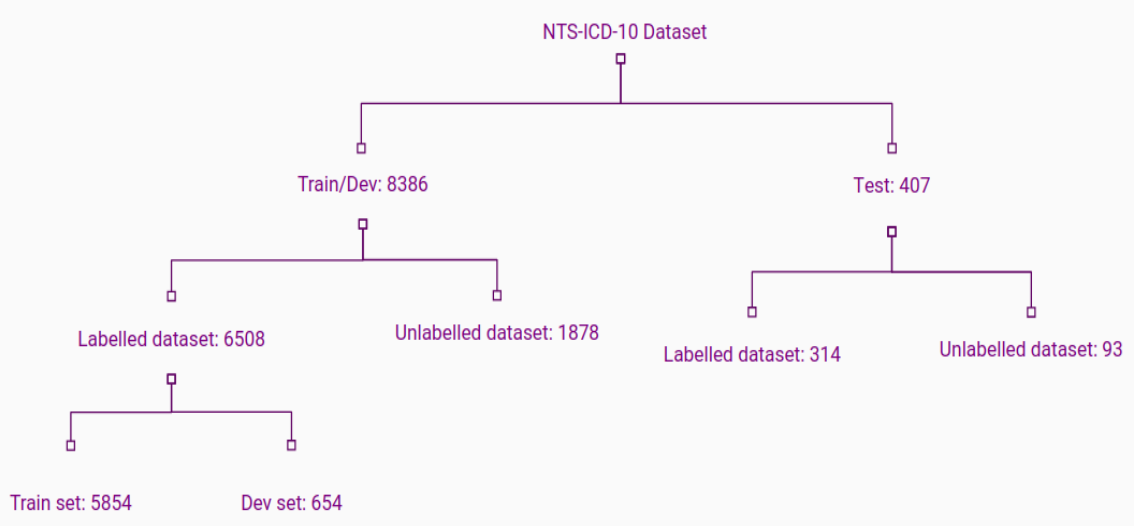


Figure 3.1: Number of training, development and test documents in the evaluation dataset-(NTS-ICD-10).

3.1 ICD-10

In this section, we will discuss the meaning of *ICD-10*, the representation of Chapters and groups, the sub-groups inside each main group which leads down to the *main diagnosis code* which makes it look like a hierarchy. We will also look into what each *ICD-10* Chapters and group actually represent in case of classification of diseases.

ICD-10 according to Geneva (2019) is a classification of diseases that can be defined as a system of categories to which morbid entities are assigned according to established criteria. The purpose of *ICD-10* is to permit the systematic recording analysis, to translate diagnoses of diseases and other health problems from words into an alphanumeric code, which permits easy storage, retrieval and analysis of the data (Geneva 2019). It can be used to classify diseases and other health problems recorded on many types of health and vital records. It is also an organized method which is used to classify diseases or the injuries behind it which has been its main purpose. As already discussed that *ICD-10* is used for classification of diseases using some alphanumeric code, it divides the whole classification into *22 Chapters* with *roman numbers* which is further divided into *certain groups of 6 characters*.

Table 3.1 shows the *Chapters* and its associated *groups* belonging to *ICD-10* as proposed by WHO (2019). Each Chapter and groups are associated with specific group of diseases, and it is better described in the title section of the table.

Table 3.1: ICD-10 Chapters present in ICD-10 tree (Based on WHO 2019)

Chapter	Groups	Title
I	A00-B99	Bestimmte infektiöse und parasitäre Krankheiten
II	C00-D48	Neubildungen
III	D50-D90	Krankheiten des Blutes und der blutbildungen Organe sowie bestimmte Störungen mit Beteiligung des Immunsystems
IV	E00-E90	Endokrine, Ernährungs- und Stoffwechselkrankheiten
V	F00-F99	Psychische und Verhaltensstörungen
VI	G00-G99	Krankheiten des Nervensystems
VII	H00-H59	Krankheiten des Auges und der Augenanhangsgebilde
VIII	H60-H95	Krankheiten des Ohres und des Warzenfortsatzes
IX	I00-I99	Krankheiten des Kreislaufsystems
X	J00-J99	Krankheiten des Atmungssystems
XI	K00-K93	Krankheiten des Verdauungssystems
XII	L00-L99	Krankheiten der Haut und der Unterhaut
XIII	M00-M99	Krankheiten des Muskel-Skelett-Systems und des Bindegewebes
XIV	N00-N99	Krankheiten des Urogenitalsystems
XV	O00-O99	Schwangerschaft, Geburt und Wochenbett
XVI	P00-P96	Bestimmte Zustände, die ihren Ursprung in der Perinatalperiode haben
XVII	Q00-Q99	Angeborene Fehilbildungen, Deformitäten und Chromosomenanomalien
XVIII	R00-R99	Symptomme und abnorme klinische und Laborbefunde, die anderenorts nicht klassifiziert sind
XIX	S00-T98	Verletzungen, Vergiftungen und bestimmte andere Folgen äußerer Ursachen
XX	V01-Y84	Äußere Ursachen von Morbidität und Mortalität
XXI	Z00-Z99	Faktoren, die den Gesund Gesundheitzustand beeinflussen und zur Inanspruchnahme des Gesundheitswesens führen
XXII	U00-U99	Schlüsselnummern für besondere Zwecke

Chapter I here itself is characterized as a group between *A00-B99* and it represents diseases that are infectious and parasitic, *Chapter VI* contains all the diseases that are related to the nervous systems and is characterized as a group between *G00-G99*, and so on. We will have a deeper look inside the organization of groups until we reach to the *main diagnosis code* which we can also call as the *leaf node* of the *ICD-10* tree[4].

[4]ICD10-tree https://www.dimdi.de/static/de/klassifikationen/icd/icd-10-gm/kode-suche/htmlgm2019/index.htm

The groups *(A00-B99, C00-D48, G00-G99, M00-M99)* seen in the Table 3.1 are represented as the top level groups of the *ICD-10* tree which we can also refer to as a parent or ancestor. And, then each parent has several childrens which forms several layers. Like in *A00-B99*, there is only single level of sub-groups or childrens which are siblings of each other. But, to explain some layered architecture, the labels hierarchy like *C00-C97* $\rightarrow$ *C00-C75* $\rightarrow$ (*C00-C14, C15-C26, ...*) means that *C00-C97* is the parent of *C00-C75*, which again is the parent of *C00-C14* and its siblings. This makes it a 3-layered hierarchy. Likewise, the group *M00-M99* consists of only a 2-layered hierarchy in the *ICD-10* tree. These are not the final level in the hierarchy, but only in the ones having *group level* codes which further leads to three and four character *main diagnosis codes*.

Since we heard of the *hierarchy* here, it is now better to see how the groups in the hierarchy actually looks like which would make it easier to understand the representation in the *main diagnosis code*.

Table 3.2: Chapter I (A00-B99) sub-groups in ICD-10 tree (Based on WHO 2019)

Groups	Title
A00-A09	Infektiöse Darmkrankheiten
A15-A19	Tuberkulose
A20-A28	Bestimmte bakterielle Zoonosen
A30-A49	Sonstige bakterielle Krankheiten
...	...
...	...
B95-B98	Bakterien, Viren und sonstige Infektionserreger als Ursache von Krankheiten, die in anderen Kapiteln klassifiziert sind
B99-B99	Sonstige Infektionskrankheiten

Table 3.2 shows the sub-groups associated with *Chapter I groups* which only has one level of hierarchy excluding the leaf nodes as mentioned above. Every sub-group now has a set of three and four charactered codes which is also represented as a *leaf node* or a *main diagnosis code*. As an example of these, in the group *A00-A09*, we have a three character code *(A00.-)* which represents different types of diseases related to *cholera*. But, still this is not the *main diagnosis code* which can be differentiated by a *dagger (-)* symbol at the end of the code (WHO 2019). The code *A00.-* further contains *main diagnosis codes* as seen in Table 3.3:

Table 3.3: Main diagnosis codes in A00.- dagger section in ICD-10 tree (Based on WHO 2019)

Code	Title
A00.0	Cholera durch Vibrio cholerae O:1, Biovar cholerae
A00.1	Cholera durch Vibrio cholerae O:1, Biovar eltor
A00.9	Cholera, nicht näher bezeichnet

Each Chapter in the *ICD-10* tree is a representation of some disease groups. **Chapter I** is about the diseases that are generally recognized as contagious or communicable. These type of diseases are infectious and parasitic which can result in diseases like tuberculosis(A15-A19), virus infections in *Central Nervous System (CNS)* (A80-A89). And, similarly **Chapter II** is about *neoplasms* which can be of four different types as *malignant*(C00-C97), a tumor which can lead to be cancerous and threatening if untreated, *benign*(D10-D36), a tumor less threatening than *malignant* and non-cancerous, *in-situ*(D00-D09), a tumor which is on the spreading state and can result in *malignant*, and *neoplasms of unknown behavior*(D37-D48), a tumor which has unsafe behavior meaning no known site of the tumor and cannot be classified as whether *malignant* or *benign* which we can see in Figure 3.2.

As we saw the *groups* from *Chapter I*, we can tell it has a 3 level of hierarchy. Next, we will see the *groups* from *Chapter II* where we can see a 5 level of hierarchies until we reach down to the *main diagnosis codes*. Figure 3.2 shows the grouping of *ICD-10* groups belonging to *Chapter II*. So, the process of diagnosing a patient with a *Nasenhöhle* disease will be simplified as *C00-C97* $\rightarrow$ *C00-C75* $\rightarrow$ *C30-C39* $\rightarrow$ *C30.-* $\rightarrow$ *C30.0* where the first three are the group levels being simplified down to the *dagger* level code which further goes to the final classification of the disease. *Chapter II* has the most number of group levels whereas the Chapters having upto four level of hierarchies are *Chapter XIII and XIX*. So, if a patient has to be diagnosed with *Flachrücken*, it would be simplified as *M00-M99* $\rightarrow$ *M40-M54* $\rightarrow$ *M40-M43* $\rightarrow$ *M40.3-* and then use one of the digits from (0-9) to represent the fifth digit in the code to create a final classification code as mentioned in WHO (2019). All these group levels and codes represent the ones from the *ICD-10* tree, which has the maximum level of hierarchy of 5 as it contains the codes until the *main diagnosis code* or the *final classification*. But, in the *evaluation NTS* dataset, we do not have the code until the last level, but only until the group before it reaches the *dagger code* which results in the maximum level of hierarchy of 3 which results as *C00-C97* $\rightarrow$ *C00-C75* $\rightarrow$ *C30-C39* for a patient with *Nasenhöhle* disease.

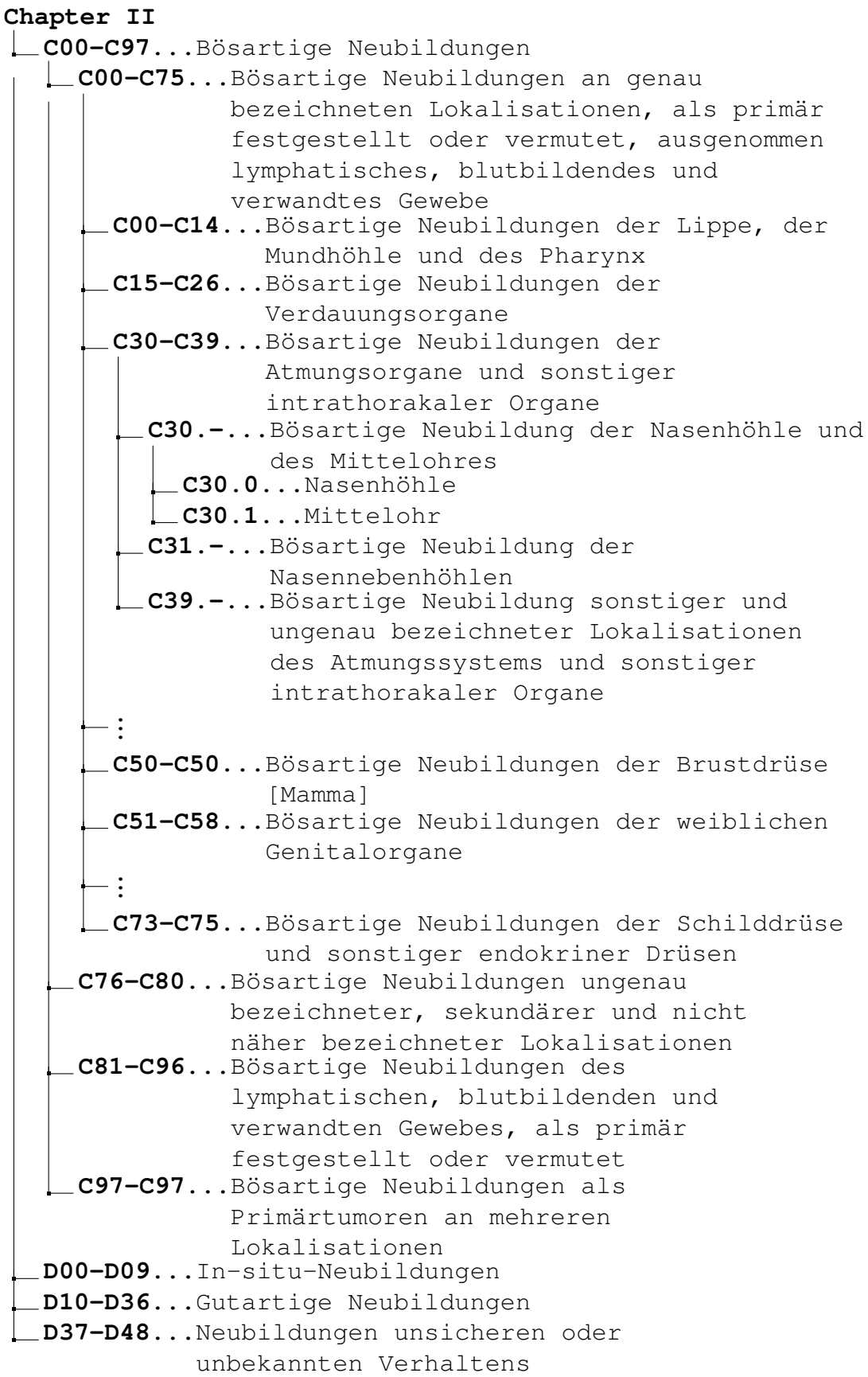


Figure 3.2: Chapter II (C00-D48) sub-groups present in ICD-10 tree (Based on WHO 2019)

3.2 Non-Technical Summaries description

NTS is a document containing the brief description of the work or experiment being conducted along with the necessary findings if any. *NTS* is a concise document that provides a description of the underlying assessment process and its findings in a manner that is both appealing to read and easily understood by the general public (Murphy 2012). As mentioned that our *evaluation* dataset is about *NTS* of *animal experiments*, it provides a clear and detailed information about the purpose, benefits, and the careful handling of the animals involved. As a result of these experiments, each *NTS* is assigned with an *ICD-10* code based on the output of the experiments while there are 1878 *NTS* documents that do not have any *ICD-10* codes assigned as seen in Figure 3.1.

Keeping in mind about the proper handling of the animals involved in the experiment, Russell and Burch (1959) provides with the principle of *Three Rs (Replacement, Reduction, Refinement (3R))*. The development of *3R* principle in these types of experiment areas may be most efficient, as a large number of experimental animals would benefit from it (Bert et al. 2017). Finally, the *NTS* document used in our *evaluation* dataset mentions about the purpose of the experiment, benefits that would be gained, or the stress or injury that might happen along with the measures mentioned in the *3R* principle.

So, to be specific, each *NTS* document of our *evaluation* dataset contains 6 fields as:

1. Title of the document
2. Uses(Goals) of the experiment
3. Possible harms caused to the animals
4. Comments about replacement (in the scope of the *3R* principle)
5. Comments about reduction (in the scope of the *3R* principle)
6. Comments about refinement (in the scope of the *3R* principle)

In Figure 3.3, we can see the the statistics and length of all the different fields available in the *NTS* documents. Also, out of all the 6 fields, the *goals* section is more lengthy and hence also containing more number of words in average. This implies that this field also contains the main detail about the experiment which is highly valuable to the model. We can see an example content of a sample *NTS* document below where we can see that the specific terms valuable for a classification model are in *title* and *goals* whereas the rest are more descriptive about the conduct of the experiment

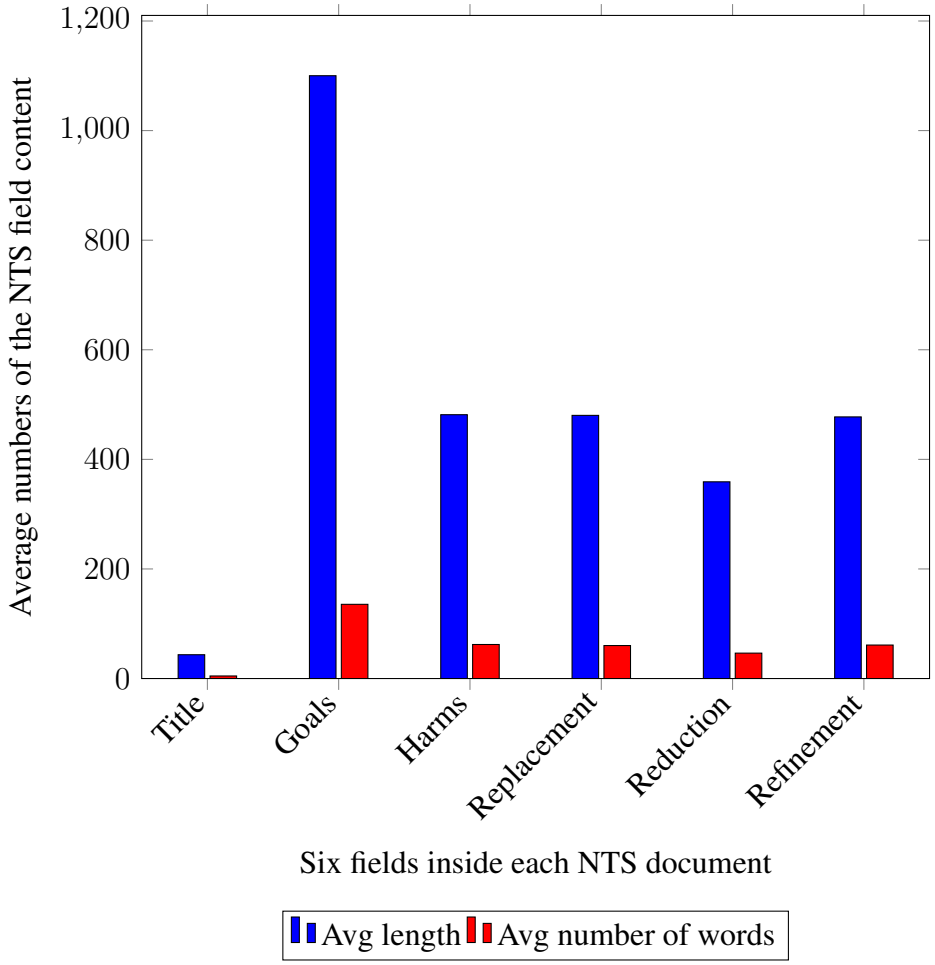


Figure 3.3: Statistical values of 6 fields content in NTS training documents

We will get a better overview of all these 6 fields by simply looking inside one of the *NTS* document from our *evaluation* dataset.

> **Field 1: Title of the documet**
> Therapieansätze f metastastasierenden **Krebs**

Field 1 of the *NTS* document is about the title of the experiment which here is to see about the *therapeutic approaches for metastatic cancer*. This simply tells us that this experiment is related to a study about cancer and its preventive or treatment measures.

> **Field 2: Uses(Goals) of the experiment**
> Im Rahmen dieses Tierversuchsvorhabens soll mit Hilfe von Mäusen das **Tumorwachstum** und Metatasierungsverhalten unterschiedlicher **Brust-** und **Eierstockkrebsarten** untersucht werden. Darüber hinaus ist mit Hilfe neuer therapeutischer Testansätze, die in der Zellkultur für diese **Krebsarten** erfolgversprechende Resultate geliefert haben, ein in vivo Ansatz als weiterführender Forschungsschritt notwendig, um eine **tumor**-therapeutische Wirksamkeit auch im lebenden Organismus zu überprüfen. Die Therapeutika sollen den Tieren oral verabreicht werden, um die Belastung der

> Mäuse so gering wie möglich zu halten. Erkenntnisse aus diesem Projekt leisten einen großen Beitrag zum Verständnis von Tumorwachstum und der Tumorstreuung (Metastasierung) im Körper, sowie zur Reaktion der Tumorzellen auf die hier verwendeten Therapieansätze. Alle hier vorgesehenen Versuche werden auf der Grundlage guter wissenschaftlicher Praxis durchgeführt.

Field 2 of the *NTS* document is about the goals of the experiment. The text in this field briefly explains about different types of *tumor growth* and *cancer* species that will be conducted with the help of mice. And, it also further investigates on the *therapeutic* test approaches to measure the result of the behavior in the *cell culture* for this *cancers*.

> **Field 3: Possible harms caused to the animals**
> Während dieses Tierversuchsvorhabens werden den Mäusen einmalig **Tumorzellen** oder Kontrollzellen unter die Fellhaut injiziert. Sobald bei den Tieren Schmerzen oder Leiden erkennbar sind, werden sie tierschutzgerecht getötet.

Field 3 of the *NTS* document explains about the harms that could be caused to the animals during the experiment. And, in case the pain is severe, the experimented animals are killed in a less painful way which is appropriate to the welfare of the animals.

> **Field 4: Comments about replacement (in the scope of *3R* principle)**
> Die Limitation für weiterführende Erkenntnisse in diesem Forschungsvorhaben ist die Testung tumortherapeutischer Wirkungen in einem gesamtphysiologischen Zusammenhang, weshalb diese in vivo Versuche unerlässlich sind, da eine mögliche klinische Relevanz der Ergebnisse für betroffene **Tumor**patienten nicht durch Zellkulturexperimente erreicht werden kann.

After the first 3 fields, each *NTS* document is followed by the measures of *3R* principle. **Field 4** of the *NTS* document explains about some alternative approaches for the experiment if possible which does not have to use animals. This replacement could also be about using only the cell cultures instead of whole animals because it is the cell or the microorganisms that are to be cleaned (Russell and Burch 1959).

> **Field 5: Comments about reduction (in the scope of *3R* principle)**
> Auf der Basis standardisierter Versuchsprotokolle und der Ergebnisse aus den vorangegangenen in vitro Testungen wurde dieses Tierversuchsvorhaben auf ein Minimum reduziert, indem zu testende Therapeutika, die schon in vitro keine übermässige therapeutische Wirksamkeit erzielt haben, bereits für einen Tierversuch ausgeschlossen wurden. Die dadurch erzielte minimal notwendige Anzahl von Tieren ist erforderlich, um individuelle Schwankungen innerhalb der Versuchsgruppen zu minimieren.

Field 5 explains about the correct number of animals to be used for experiments by choosing the right choice of strategies in the planning and performance of the whole experiment (Russell and Burch 1959). While this field also stresses more on reducing the use of animals for experiments but, also keeping the number of animals sufficient to obtain a statistically significant outcome of the experiment.

> **Field 6: Comments about refinement (in the scope of *3R* principle)**
> Aus vorangeganenen Studien ist der hier verwendete Mausstamm optimal geeignet für das Versuchsvorhaben. Die initiale Überwachung der Tiere während des Versuchsablaufs soll durch Erhöhung der Überwachungsfrequenz nach Tumorentstehung und entsprechendem Therapiebeginn unerwartete Tierbelastungen oder Unwohlsein frühzeitig erkennen lassen. Mit mehrjährig erfahrenem Personal und anhand der stringenten Abbruchkriten und engmaschigen Kontrollen werden Tierbelastungen so gering wie möglich gehalten.

Field 6 mentions about the *refinement* procedure but with respect to both *replacement* and *reduction* measures. It is about giving proper care to the animals being tested by giving better living conditions or reducing the pain during experiments which might cause instability in the experiments. After achieving this, *refinement* might improve the data quality and could be of some help to the *reduction* measure.

As mentioned earlier, *NTS* documents from the *evaluation* dataset is assigned with its respective *ICD-10* codes based on the output of the experiments, the *NTS* document that we just discussed is assigned with *ICD-10* codes like *II, C00-C97, C00-C75, C51-C58, C50-C50*. Let us also keep a note of the *highlighted words* above like **Krebs, Brustkrebs, Eierstockkrebsarten, Tumorzellen, Krebsarten** which plays an important role to grab the attention of the underlying language model. As we noticed that there are no any *three or four* character codes or the *main diagnosis code* in the *ICD-10* codes assignment for this document, let us take the lower-level group which would be *C50-C50 and C51-C58*. And, then referring to the Figure 3.2, we find that *C50-C50* represents **Brustkrebs** and *C51-C58* represents **Krebs in Genitalorgane** which in this case is **Eierstockkrebsarten**. And, the remaining codes *II, C00-C97, C00-C75* would be the automatic assignment with respect to other codes as they are parent of each other in the *ICD-10* tree for *Chapter II* as represented in Figure 3.2.

The document described above is one example of *NTS* document in our *evaluation* dataset, and every document has all those six fields available in it. The goal of the experiment is to test different treatment approaches and to come up with a better and effective solution of performing experiments by also utilizing fewer animals for experiments and in a less painful way.

3.3 Code Statistics

In this section, we will go deeper into our *evaluation* dataset containing *NTS* documents and perform some evaluations or analysis on the *ICD-10* codes assigned to each documents like *most frequent codes* which would also tell us about the most occuring disease. This dataset has a variety of code assignments like the documents are either only assigned with *Chapter codes*, *group codes* or both.

As mentioned earlier in Section 3.2, the *NTS* documents from the *evaluation* dataset is assigned with one or more *ICD-10-GM* codes which exhibits the property of a *multi-label* classification. The top level of the *ICD* tree is the *Chapter* which then is followed by its respective *groups* which maintains a hierarchial structure as seen in Figure 3.2. The *NTS* documents which are labelled as seen in Figure 3.1 are assigned with either a single code which only contains a *Chapter ICD-10 code* or a group of codes which contains both *Chapter or group from the ICD-10 code*.

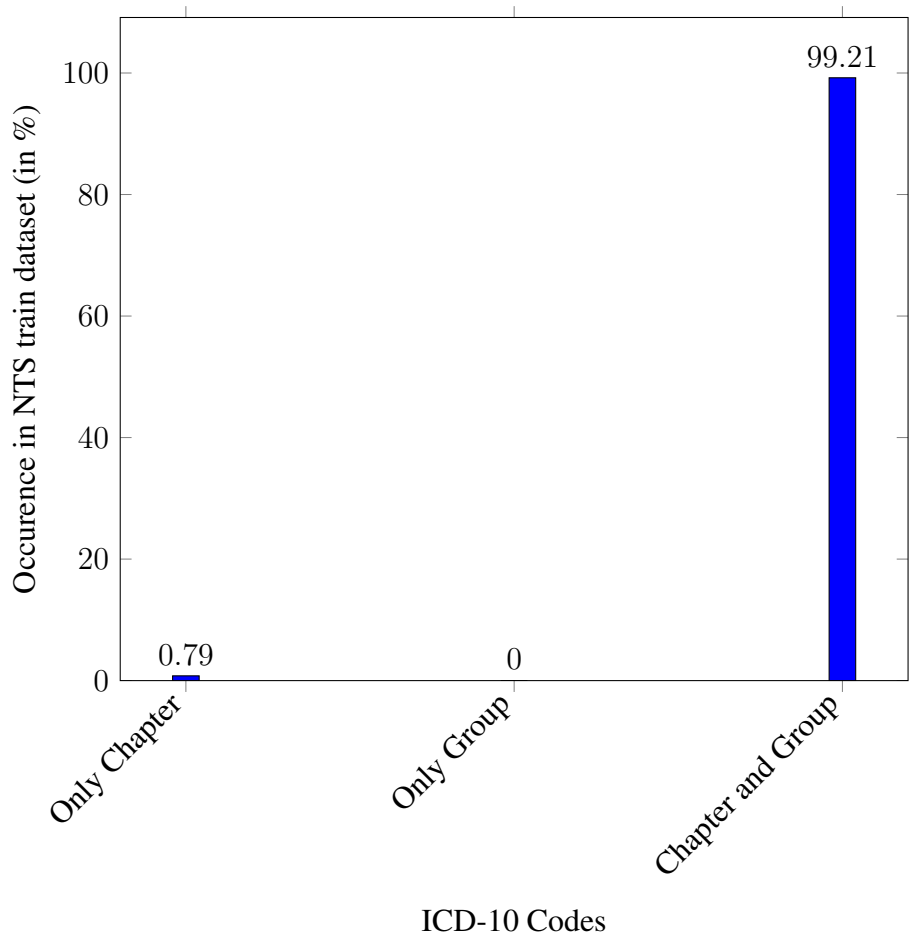


Figure 3.4: Types of ICD-10 codes in NTS train documents

In Figure 3.4, we can see that 99% of the *NTS* documents are assigned with both *Chapter and group ICD-10* codes together, while we cannot see any documents with only labelled with *group ICD-10* code. Figure 3.5 shows the *ICD-10* codes associated with *Chapter*

II, and also supports the statement "*most of the NTS documents have Chapter and group occuring together*". We choose *Chapter II ICD-10* codes for this demonstration as the codes from *Chapter II* are the most occuring ones in our *evaluation* dataset as seen in Figure 3.6.

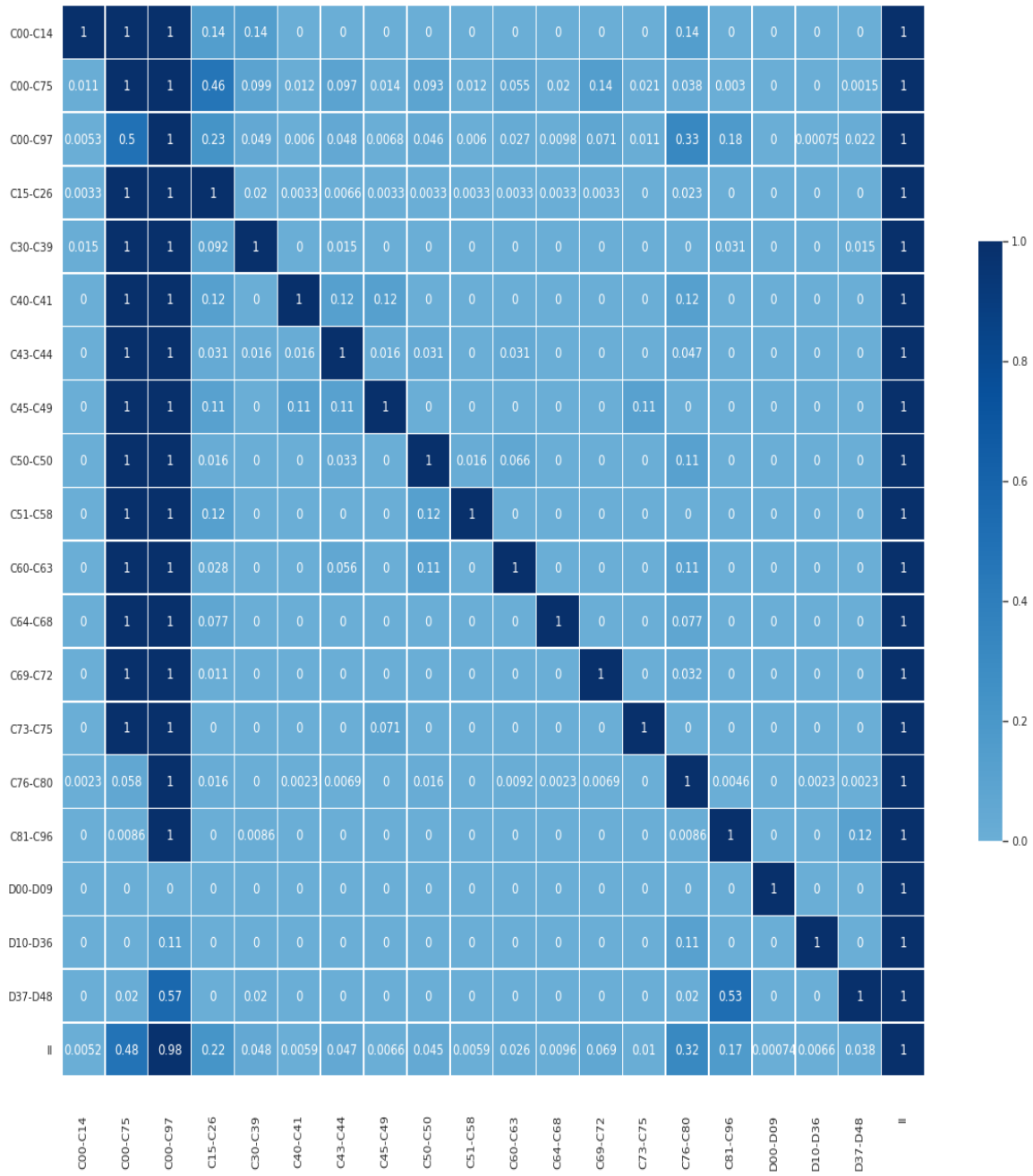


Figure 3.5: Conditional occurence of ICD-10 codes in Chapter II and its groups in NTS training dataset.

Figure 3.5 shows the conditional probability ($P(y|x)$) representation of the occurrence of a group while the other group is present for the *ICD-10* Chapter *II* codes, where y is the *ICD-10* group to occur in the presence of another *ICD-10* group, here represented by x, and hence the combination (y, x) to browse the probability table. So, the x and y in the

figure represents the *x-axis* and *y-axis* respectively.

So, keeping that in mind, if we see the occurrence of the groups in *y-axis* with respect to *Chapter II* in *x-axis*, we will see *1*. This association strongly tells us that with every *child group* in the *NTS* document, it also consists of its *Chapter*. Besides *Chapter and group*, we also have another representation to support the *hierarchial* representation of the *ICD-10* tree. For example, if we consider the first row *C00-C14* from Figure 3.5, and the parents of it from Figure 3.2 which are *C00-C75, C00-C97 and II*, it tells us that if a child group is present as an *ICD-10* code in the document, it is also accompanied by its respective parent *ICD-10* group code. This is verfied with the cell value of *1* in the combination of *(C00-C14, C00-C75), (C00-C14, C00-C97), and (C00-C14, II)* in Figure 3.5.

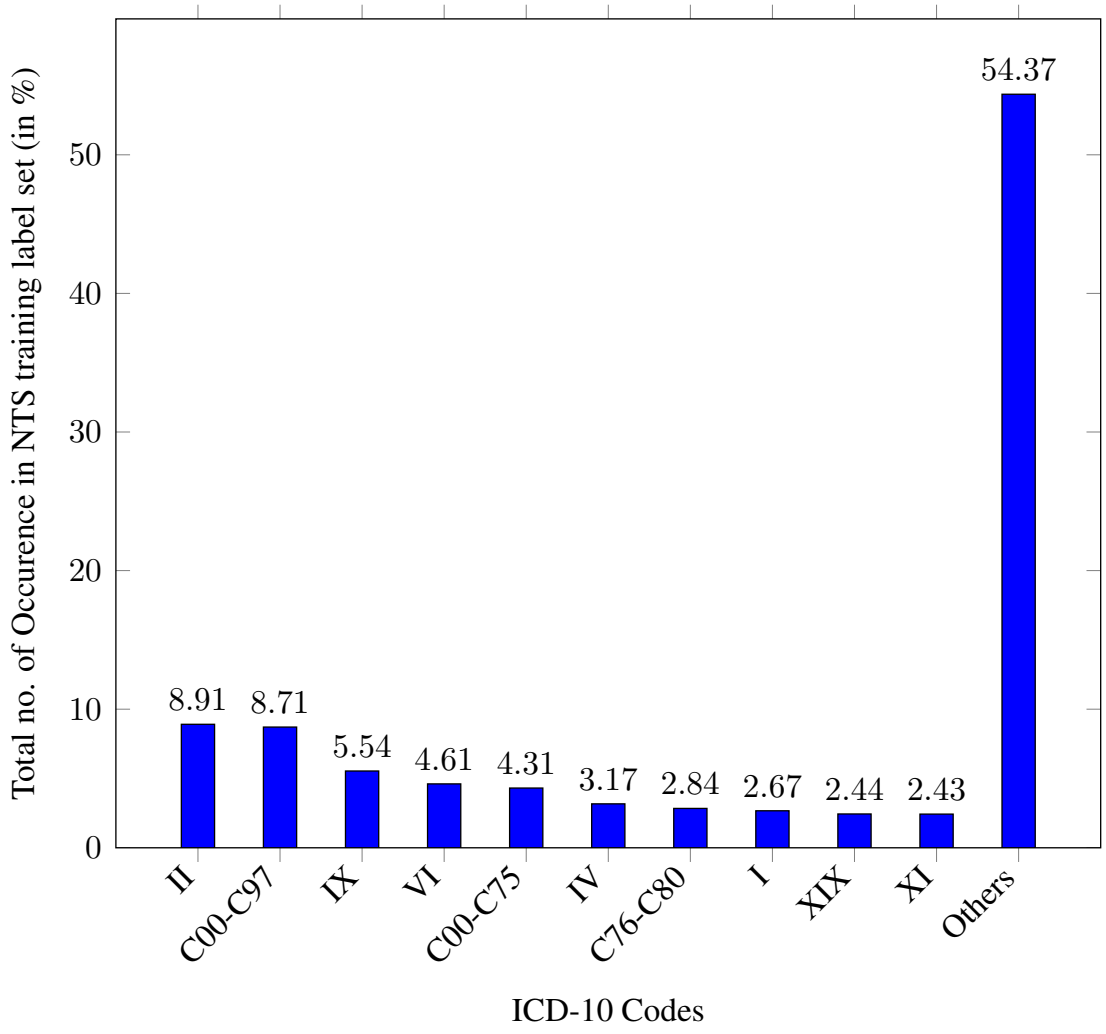


Figure 3.6: Frequent ICD-10 codes in NTS training dataset

Figure 3.6 shows top-10 most frequent codes available in the *NTS-ICD* training dataset. It also shows that most frequent code was from *Chapter II* followed by the group C00-C97 which tells us that *Neoplasms* was the dominating diagnosis among all and is also supported by Figure 3.8.

As mentioned earlier that the *NTS* documents are associated with their respective *ICD-10* codes depending on the result of the experiment, there is a variable number of code assignment for the *NTS* documents. Figure 3.7 shows the distribution of codes per document in training set. Number of *NTS* documents having 2 codes are 64.6% out of all the *NTS* documents available in the training set. Around 0.75% of the training documents have only 1 code in it, and it is the *Chapter* code from the *ICD-10* tree as seen in Figure 3.4. The difference in the number of documents having only *Chapter* codes in Figure 3.4 and 3.7 is that the later considers documents having only 1 *Chapter* code per document while the former case considers all documents having *Chapter* codes which tells us that there are *NTS* documents having more than one *Chapter* code. Figure 3.7 also shows that the variability of number of code in the documents ranges from 1 to 12 where there are no documents which are labelled with 11 codes, while the highest number of code 12 is assigned to only one document with *(C00-C97, J09-J18, L20-L30, I, II, X, T89-T89, XII, C76-C80, XIX, A30-A49, B50-B64)*.

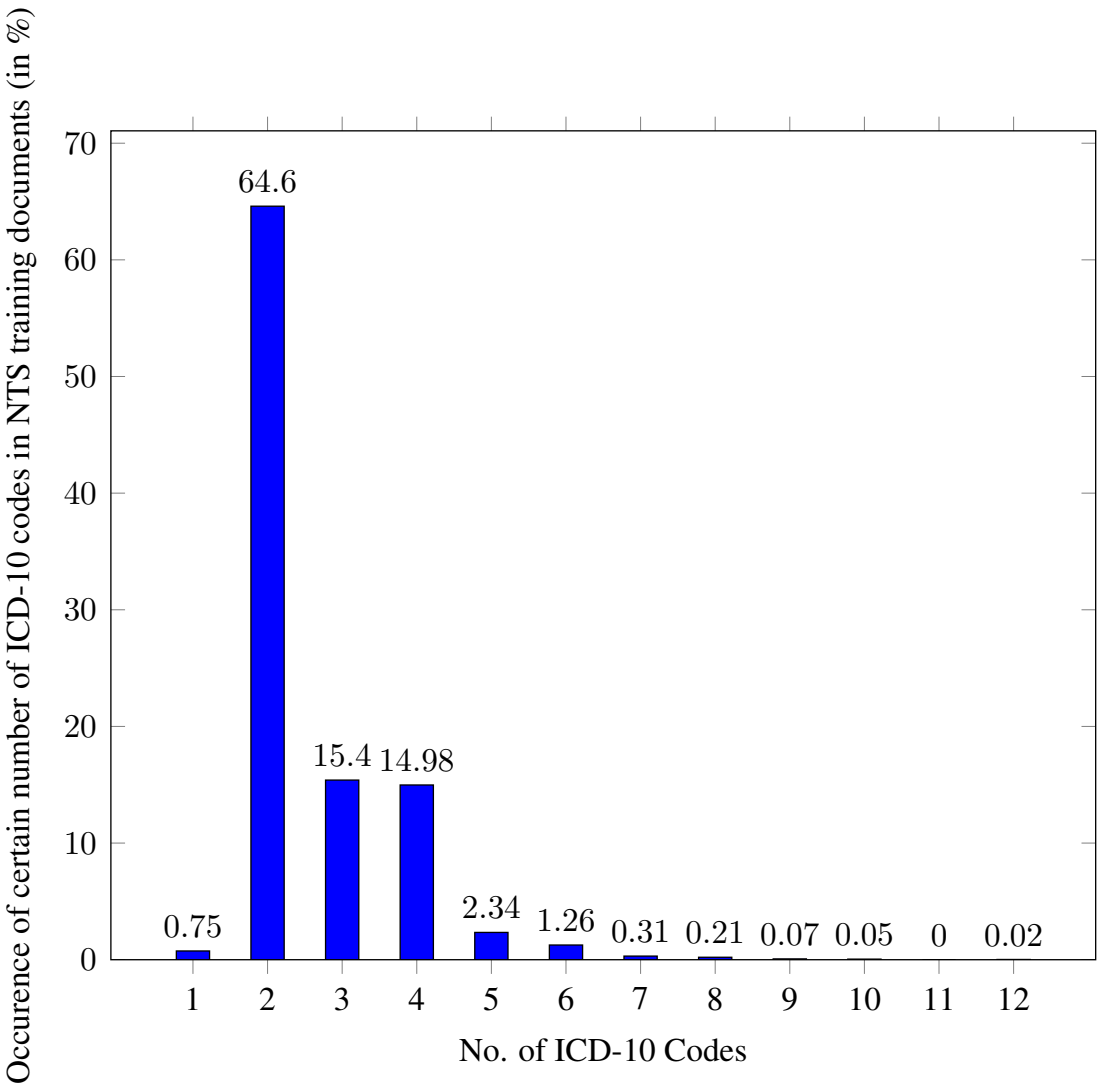


Figure 3.7: Number of NTS training dataset consisting of certain range of ICD-10 codes in it

Figure 3.8 shows the distribution of Chapter code occurence in *NTS* training documents. As we can see Chapter II has the highest number of occurence, i.e. 8.91%, it also resembles that the dominant group set is C00-C97 as seen in Figure 3.9. This figure also gives us the information that the dataset mostly contain experiments related to neoplasms or cancer. Diseases of the *neoplasms (Chapter II)*, diseases of the *circulatory system (Chapter IX)*, and diseases of the *nervous sytems (Chapter VI)* as seen in Figure 3.8 has been the most occuring one in the available experiments.

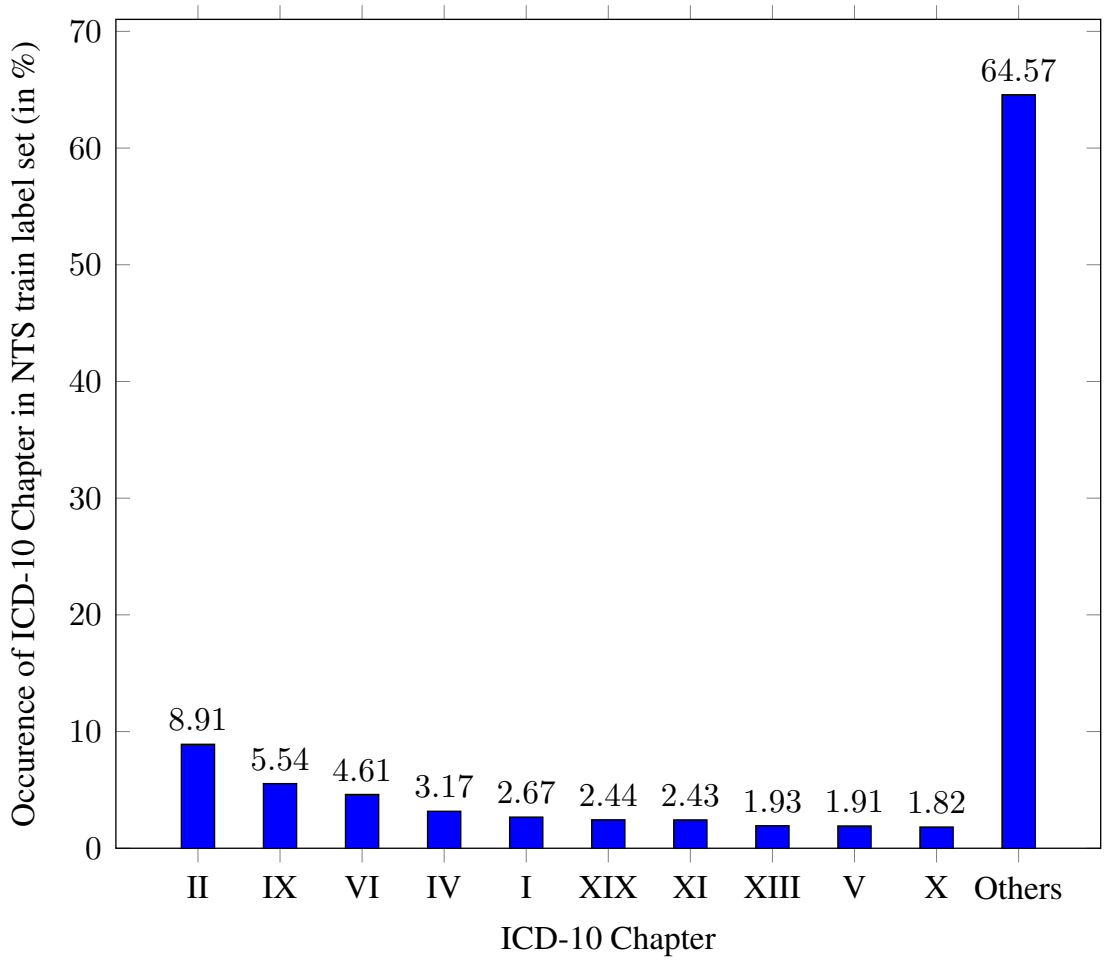


Figure 3.8: ICD-10 Chapter Occurence count in NTS train label set

For example, *Malignant neoplasms in specified locations, identified or suspected as primary, excluding lymphoid, hematopoietic, and related tissue (C00-C75)* are mostly allocated to NTS in the field of neoplasms followed by *Malignant neoplasms of inaccurately identified, secondary and unspecified locations (C76-C80)*. Other forms of cancer *Malignant neoplasms of digestive organs (C15-C26)* also rank high in NTS.

As we can see in 3.4, most of the document in the *NTS* dataset is assigned with a corresponding Chapter with their respective group. And, the most occuring Chapter is *II* which tells us that *neoplasms*, which is the identifcation for Chapter *II* is the dominating disease group in the dataset.

Likewise, Figure 3.9 shows the distribution of group code occurence among the *NTS* training documents. As mentioned in Figure 3.8, Chapter II is the most frequent Chapter, and group C00-C97 is the most frequent group which is followed by C00-C75, C76-C80, C15-C26 which makes a further resemblance that the dataset mostly contains experiments about cancer or neoplasms. The second most frequent group is I30-I52 which is described as *other forms of heart disease*, and is under Chapter IX which is described as *diseases of the circulatory system*. This tells us that *neoplasms* and *circulatory system* diseases are the most common and with high risks.

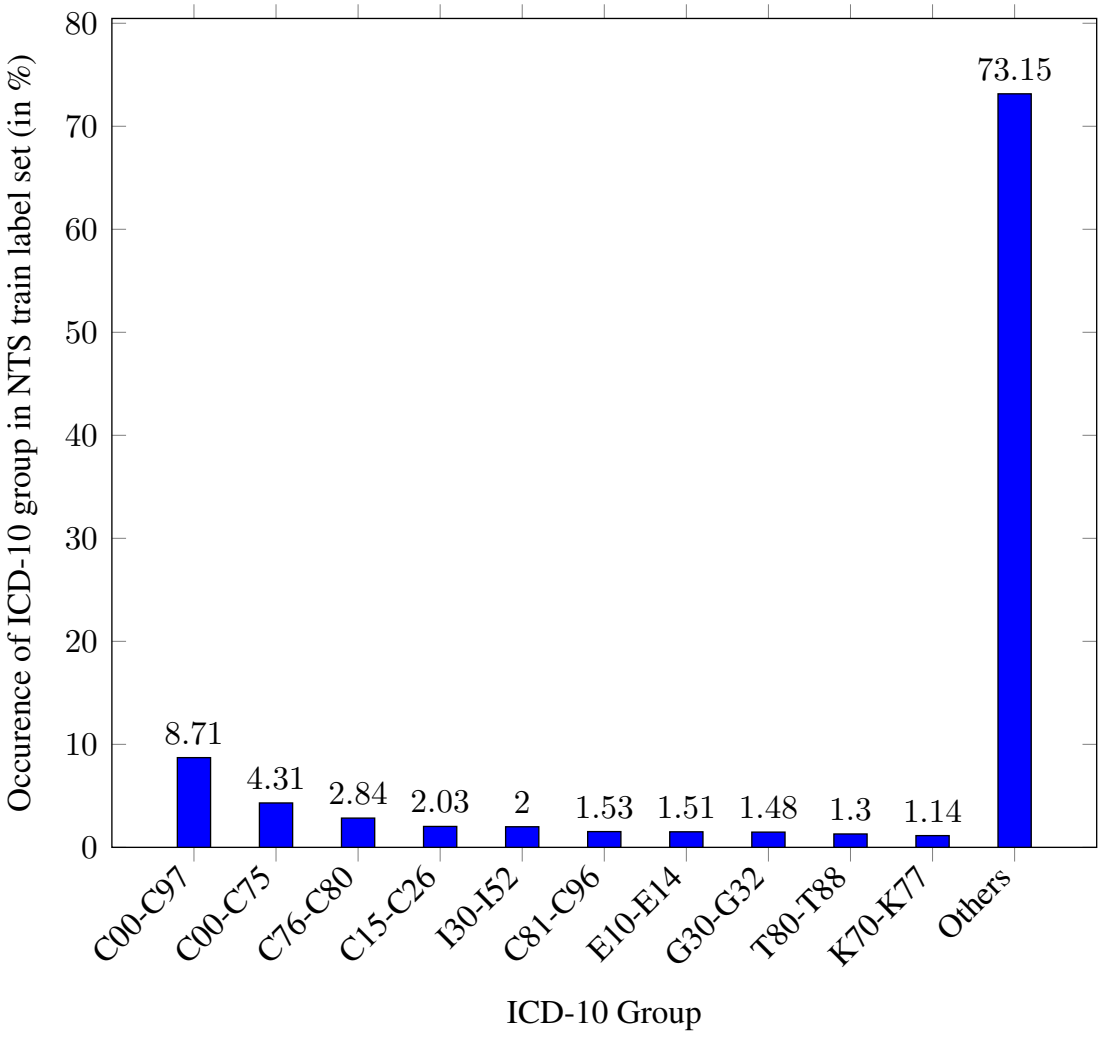


Figure 3.9: ICD-10 Group Occurence count in NTS train label set

3.4 Dataset for Fine-tuning

As mentioned earlier in the beginning of chapter 3, we are using two datasets, *evaluation dataset* using the *NTS-ICD-10* dataset to evaluate the model's performance after *fine-tuning* it using the dataset that we collected from different sources in the web. So, we collected various medical articles related to *disease, symptoms* that consisted of *medical*

terms that were not present and not learned by the *German BERT* model.

For the existing model to handle medical terminologies we collected medical datasets by crawling different medical forums, websites where we could find medical articles in German language and *fine-tuned* it. We collected the medical articles from the articles listed in Figure 3.10.

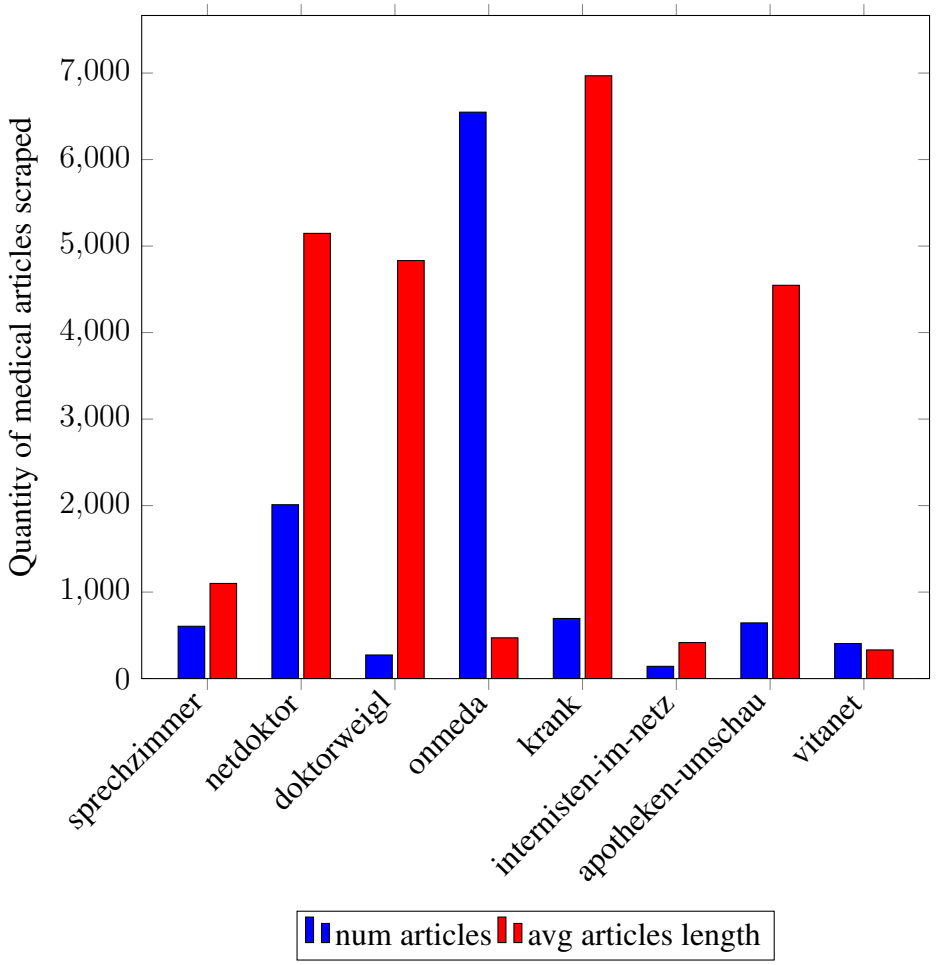


Figure 3.10: Articles scraped from medical websites to create fine-tuning dataset

Some of the articles content were much lengthier than others containing more medical terms as seen in Figure 3.10. Articles from *netdoktor, doktorweigl, krank, and apotheken-umschau* contained more sections in the articles like *medikamente, symptome, diagnose* and had more words which contained medical terms compared to other sources like *onmeda*, which was a forum to discuss for medical related problems.

This dataset thus collected needed some processing and we arranged it as a separate dataset in the following ways:

- Sentence dataset from all the combined articles.
- Segment wise dataset from the sentences combined together but not exceeding the maximum sequence length of *BERT*.

As a result, the sentence dataset contained 67.5 MB of medical sentences while segments dataset contained 127 MB of medical sentences. An example for sentence dataset can be seen below:

> Die Ursache für Maltes Hautausschlag könnten Antibiotika sein.
> Doch reagiert die Haut so auf die Wirkstoffe, die eigentlich helfen sollten?
> Um dies zu verstehen, schauen wir uns kurz den Aufbau und die Funktion der Haut an.
> Diese sind mit Fasern und weiteren Zellen untereinander verbunden.
> Ebenfalls in der Epidermis liegen Nerven, welche essentiell für die Wahrnehmung von Sinneseindrücken sind (sog.

And, similary an example of a segment wise dataset can be seen below:

> Die Ursache für Maltes Hautausschlag könnten Antibiotika sein. Doch reagiert die Haut so auf die Wirkstoffe, die eigentlich helfen sollten? Um dies zu verstehen, schauen wir uns kurz den Aufbau und die Funktion der Haut an. Diese sind mit Fasern und weiteren Zellen untereinander verbunden. Ebenfalls in der Epidermis liegen Nerven, welche essentiell für die Wahrnehmung von Sinneseindrücken sind (sog.

The purpose of separating the *fine-tuning* dataset in this way is to see which dataset sample will be more better for the *BERT* language model to learn the representations for *medical terms*. Segment wise dataset is basically combining all the sentence until it reaches the maximum sequence length as supported by *BERT*.

Chapter 4

Theoretical Foundations

This chapter provides the essential background knowledge for the chapters following this thesis work. We introduce with the basic *ML* concepts, and then we focus on the neural networks which are the specific type of *ML* models used in this thesis work. Finally, we introduce the *BERT* model along with the *transfer learning* techniques which are reviewed here for the understanding of the underlying methodology. Also, we present with the learning methods such as *transformers*, *attention mechanisms* which are included within *BERT* to make it more efficient in understanding *natural language texts*.

4.1 Machine Learning

ML is a technique to automatically teach the machines to learn given its ingredient known as *sample data*. It can also be called as a subset of *AI*. It makes use of various statistical models and learning algorithms to build a trained model based on sample data known as *training data*. When the model is correctly trained, the resulting model can perform tasks like *classification, prediction, regression*, or any inferential tasks. The outputs from those tasks can later be used to perform *target tasks* like *spam detection*, *sentiment analysis*, *document classification*, and many more. The learning can further be divided into different approaches like *supervised learning*, *unsupervised learning*, *self-supervised learning*, *semi-supervised learning*, and *reinforcement learning*. We will only go through the first three approaches in the sections below as they are of specific interest in the current research.

4.1.1 Supervised Learning

Supervised learning process is the most common technique in the classification problems, where the model is trained with labeled data, i.e. a dataset consisting of both features and labels (Nasteski 2017). While each machine learning model is divided into two steps (training and testing), the training process learns the samples from the training data using a learning algorithm and build a trained model. And, as to test the efficiency of the trained model, in the testing process, the trained model is used to make the inferences for the testing data samples where the model is only fed with the features and later compare

the output with the actual labels. The most common example of a *supervised learning* problem would be *sentiment analysis* where the model is trained with a dataset containing a block of text and its respective sentiment(*positive, negative, neutral*), and tested by only feeding the trained model with a new block of text which is previously not seen by the model.

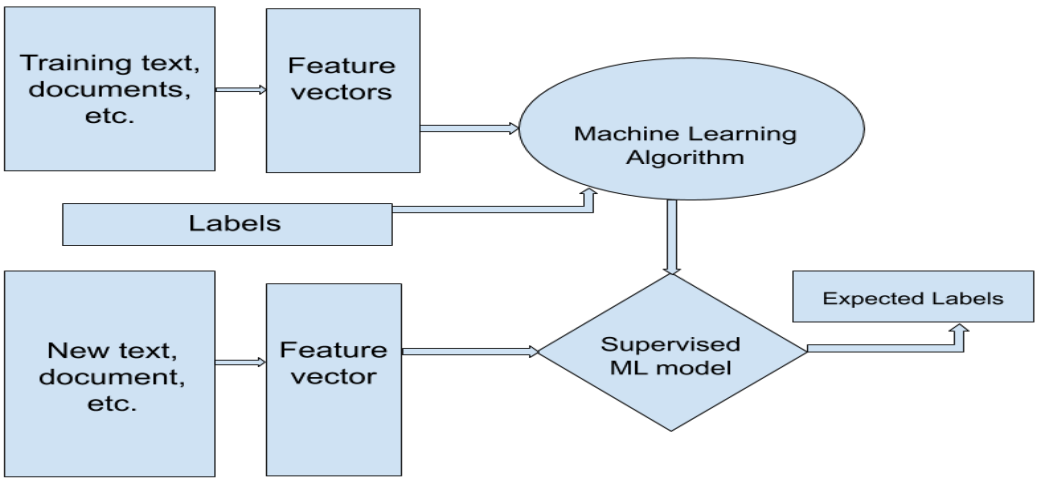


Figure 4.1: Supervised Learning Model.

Figure 4.1 shows the architecture of a basic supervised learning process. The main objective of the supervised *ML* models is to build a learning model able to predict the label of a corresponding feature object. If we refer to the *feature vectors* as X and labels as y, then the *ML* model as seen in Figure 4.1 takes the training features X and its corresponding labels y to begin the learning process and build a *trained model*. Then, this *trained model* is passed with a new set of samples X from the testing set to produce a label suitable to the features provided.

As mentioned by Nasteski (2017), supervised learning tasks are divided into two categories: *classification* and *regression*. If the labels are in a fixed set of categorical outcomes like the ones for *sentiment analysis*, then the task is called classification, but, if the labels are floating-point values like the ones in predicting the price of a house, then the task is called regression. Until this point, we found that the supervised learning process requires the training data to be labeled, so *Linear Regression, Logistic Regression, Naive Bayes* are some examples of a *ML* model to perform the supervised learning tasks.

4.1.2 Unsupervised Learning

As seen in Section 4.1.1 for supervised learning, *unsupervised learning* utilizes unlabelled data to build models that can extract the underlying structure. As this model does not make use of labeled data, this type of model has less execution time as it does not have

to annotate the input for the algorithm to learn. So, this type of learning model is more relevant for different purposes than supervised learning. The learning algorithms under this approach are mostly aimed towards *clustering*. As in the case for *NLP*, this type of learning is used to learn the word representations from a given set of textual documents and hence produce word embeddings which is later useful while performing classifications on target tasks.

4.1.3 Self-supervised Learning

This learning refers to a learning model which makes utilization of both labeled and unlabelled data. This is most useful when there is a situation of learning word representations to create embeddings and later use it in target tasks such as classification. This type of approach is already used to create language models which are therefore known as *pre-trained language model*. Some language models like *ULM-FiT, BERT* are already available which makes use of self-supervised learning. Pre-trained language models are trained with a huge amount of unlabelled text to learn contextual word representations, and hence this model is later used to perform target tasks such as *sentiment analysis, document classification* and more. This method overcomes the need for a large amount of labeled data. Hence, training using a pre-trained model is helpful to learn different contextual patterns available in a pre-trained model allowing the training of a model to be efficient.

4.2 Loss Functions

The loss function is a method to evaluate the cost of linking an event to a real number. In the context of classification problems in *ML*, it refers to linking the feature vectors to its corresponding labels. It generates the difference between the learning model's prediction and the actual labels which are further known as loss. Hence, the performance of a model is higher whenever the loss is lower which helps in selecting the desirable model. This also implies that a perfect model would have a loss value of 0. In other words, it is also known as *Objective functions* or *Cost functions* with the ultimate goal of minimizing the loss.

The loss functions differ depending on whether it is a classification task or a regression task. For classification task, it is often suitable to use the *Cross-Entropy Loss* which is also known as the *log loss*. In this topic, we will see the *cross-entropy loss* for both binary and multi-class classification. If p is the predicted probability of a class label c, M is the total number of classes, o is the observation, and y is a binary indicator that shows if a class label c is the correct classification for the observation o, then the loss equation for *binary cross entropy* and *multi-class cross entropy* would be[1]:

[1]Loss functions https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

Binary Cross-Entropy Loss: In binary classification, the number of classes *M* is 2, so the cross-entropy loss is calculated as:

$$\text{BinaryCrossEntropyLoss} = -(y\log(p) + (1-y)\log(1-p)) \tag{4.1}$$

Multi-class Cross-Entropy Loss: In multi-class classification, *M* > 2, so we calculate a separate loss for each class label per observation and sum the result as:

$$\text{MultiClassCrossEntropyLoss} = -\sum_{c=1}^{M} y_{o,c}\log(p_{o,c}) \tag{4.2}$$

As in classification problems, the classes are categorical which are later converted to binary values (0, 1), so *cross-entropy* loss is the suitable one to calculate the loss here. But, in regression tasks, the classes are some real value(floating value), so the suitable loss functions for this task are *Mean Absolute Error* or *Mean Squared Error*. This loss functions takes the predicted value respective to the target value and averages the sum of the difference between those two values which can be represented in equation as:

Mean Square Error:

$$\text{MeanSquaredError} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \acute{y}_i)^2 \tag{4.3}$$

In Equation 4.3, *n* represents the total number of samples, y is the target value, and ý is the predicted value. These loss functions discussed above are key to the learning problems of supervised classification.

4.3 Deep Learning

Deep learning is based on neural network and is a subset of *ML* methodologies which in recent years has set an exciting new trend in *ML*. It imitates the working of the human brain in processing data and creating useful patterns for use in further decision making for applications like *language understanding, speech recognition, image recognition*. The main power of *deep learning* is that it can easily learn representations from the unstructured and unlabelled data which means it has a high ability to perform *unsupervised learning*. Moreover, *deep learning* can generalize learned patterns beyond data similar to the training data, which can be advantageous while dealing with any *target task* situations (Lazreg, Goodwin, and Granmo 2016).

The word *deep* in *deep learning* refers to the larger number of layers each having a fixed size which it uses to transform the data. The number of layers in *deep learning* is usually greater than the number of layers is normal *neural network*, and thus the name *deep learning*. Each layer here is responsible to construct different features of their own. *Deep learning* as it seems to perform tasks efficiently, it also requires a large number of resources and computes time. Hence, if the *deep learning* models are given enough data, these multi-layer neural networks can automatically decompose a problem into several manageable abstractions or layers which has its feature extraction methods.

Compared to traditional *ML* methods, it reduces the need of feature extraction, and it better learns the representations with both labelled and unlabelled data. As the *deep learning* tends to generalize well, it still requires a heavy procedure to achieve high performance on various high level tasks such as *speech recognition, image recognition, language translation* and many more. In coming sections, we will see the basics of *neural network*(4.3.1), and some *deep learning* architectures like *CNN(4.3.3), RNN(4.3.4)*, and *LSTM*(4.3.4).

4.3.1 Neural Network

Neural network as being the main approach for deep learning which comprises of the larger number of hidden layers that form the network known as *deep neural network*, it differentiates itself from the *neural network* where the layer is not too dense. These networks can be trained either with supervised or unsupervised learning and hence is the basis for the capability of *deep learning*. The neural network is inspired by the working of the human brain which is designed to sense the data and recognize patterns like labeling or clustering raw input. The patterns could either be *text, images, or sound*, but for a *neural network* to process it, it must be translated into a numerical form. Being inspired by the processing and behavior of *neural networks*, many models were derived by building more complex *neural networks* with more nodes, more layers, different architectures, and mechanisms to achieve better performance in target tasks.

Structure

Figure 4.2 shows the typical structure of a *neural network* which is composed by an *input layer, hidden layer(s)*, and an *output layer*. The number of hidden layers and the amount of neurons or nodes per layer depends on the design and purpose of the network. This figure consists of only one hidden layer with three neurons. The left part of the figure consists of the input neurons of the network, and the right part is the output neurons. As mentioned earlier, the number of the output neurons depends on the application architecture. Each neuron consists of a numeric value which is passed on to the neurons of the other layer for further calculations or to the output layer to identify the final result of the network.

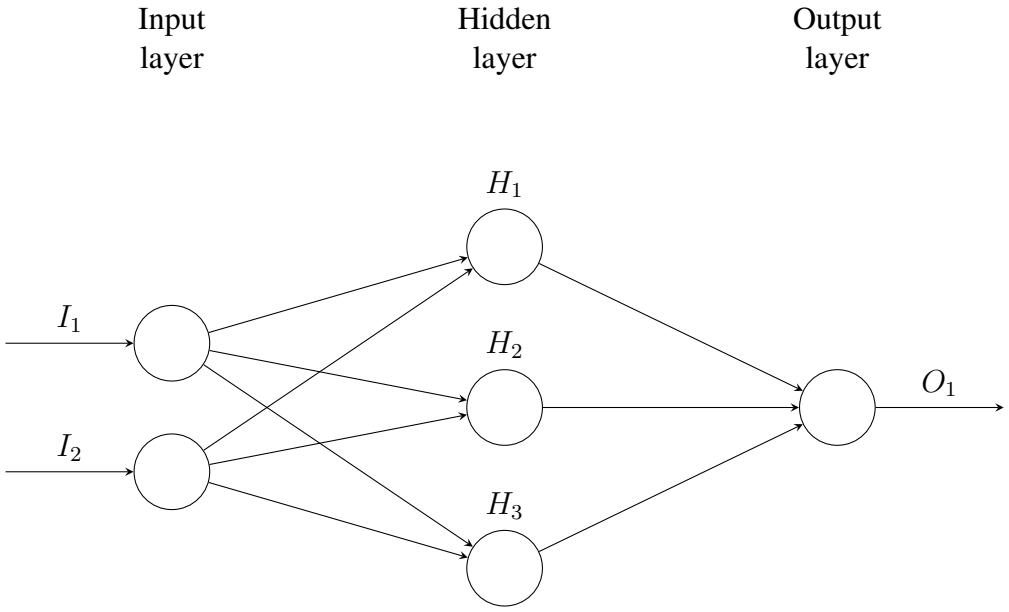


Figure 4.2: Neural Network Structure.

Neuron Output

Each neuron seen in Figure 4.2 receives one or multiple numeric values as inputs where each input has an associated weight which is assigned on the basis of its importance to other inputs. A neuron receives input from some other neurons or from an external source and computes the output.

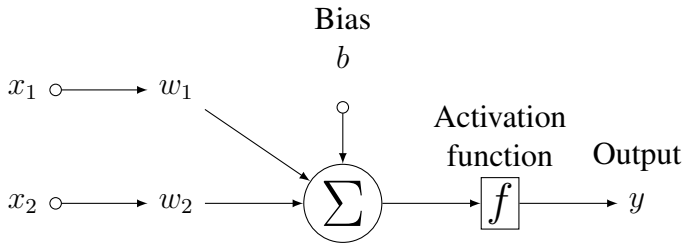


Figure 4.3: Neural Network output function.

In the Figure 4.3, the output y is expressed as the result of an activation function. The output neuron applies a function that can be seen in the figure to the weighted sum of all its inputs in addition to the bias which is calculated as:

$$y = f\Big(\sum w_{\mathrm{i}}.x_{\mathrm{i}} + b\Big) \tag{4.4}$$

The function f used in Equation 4.4 is non-linear and is called an *activation function* whose purpose is to introduce non-linearity into the output of a neuron.

Activation Functions

Activation functions as mentioned in Nwankpa et al. (2018), are functions used in neural networks to compute the weighted sum of input and biases. So, *activation functions* are useful to determine the ouput of a model, its computational efficiency, and its ability to train and converge after multiple iterations of training. As seen in Figure 4.3, the output neuron uses the values from input neurons and bias which are passed to the activation functions until the output layer gives a prediction.

Activation functions can either be linear or non-linear depending on the function it represents. The choice of an *activation function* completely depends on the nature of the considered problem. Some examples of *activation functions* used in neural network are *sigmoid, tanh, softmax*.

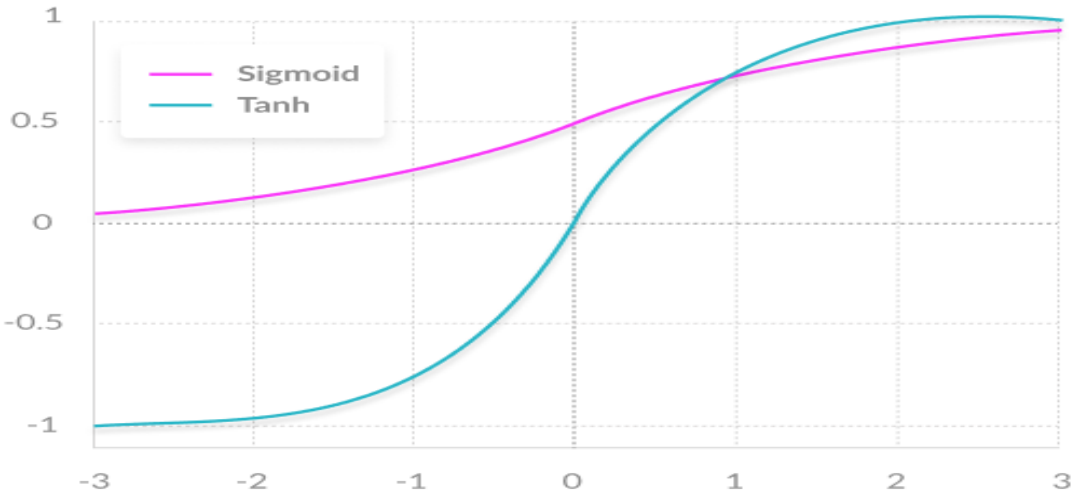


Figure 4.4: Activation functions (Taken from missinglink.ai n.d.).

Figure 4.4 shows the typical examples of *activation functions*. These activation functions for *deep learning* are non-linear functions that generally take output values between zero and one which is often easier when converting into output as a probability. Hence, this non-linearity is now at the core of neural network which is used to model complex problems as using linear activation functions could cause outputs growing towards infinities.

4.3.2 Error Backpropagation

The *back propagation* algorithm has recently emerged as one of the most efficient learning procedures for multi-layer networks of neuron-like units (Cun 1988). A forward pass is performed after a neural network is defined with initial weights to produce the initial prediction. This prediction is further evaluated using an error function which defines how different it is from the true prediction. The mechanism behind the effectiveness of this method is that it readjusts the weights of the system which allows the neurons to correct their weights in another run.

Here, *back propagation* is that method or a training algorithm for large neural networks to find a suitable weight that generates the smallest error. Utilization of this algorithm is done in four steps namely *forward pass, error function, backpropagation, weight update*. Each neural network has a phase during training which passes the input data forward through the neuron layers, from the input layer through the hidden layer(s) until it reaches the output layer, and is known as the *forward pass*. After reaching the output layer, *error function* computes the difference between the predicted value and the true value, and then this error is *back propagated* to the neurons on the network which informs about their distance to the ground truth label which allows them to correct their weights.

This process or the iteration is running until the network converges. To achieve this, various learning parameters like *learning rate, optimization method*, and more become of great help to reduce the loss function. When reaching this position, the model is ready to make predictions for any unknown data input.

4.3.3 Convolutional Neural Network

CNN is a class of *deep neural networks* which is most commonly used to analyze images. *CNN* is designed to automatically and adaptively learn spatial hierarchies of features through *backpropagation* by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers (Yamashita et al. 2018). While *CNN* is mostly suitable for analyzing images and vision, *image classification, clustering, object detection, and optical character recognition* are some of its applications.

As said before where *CNN* is constructed using three types of layers, the first two, *convolutional layer* and *pooling layer* is responsible for feature extraction, whereas the third, a *fully connected layer* is responsible for inferring the extracted features into a final output. While the *convolutional layer* and *pooling layer* are responsible for feature extraction, *convolutional layer* sequentially processes parts of the input matrix that represents an image, and *pooling layer* transforms the information into a more computationally manageable way. The *convolutional layer* and *pooling layer* of *CNN* makes the network more suited for tasks related to images by allowing the model to encode image-specific features into the architecture, and also by reducing the parameters required using compression to setup the model. *CNN* therefore allows for a simple network architecture to be set up in combination with proper choices of hyperparameters and optimization methods.

4.3.4 Recurrent Neural Network

As *CNN* was effective while processing visual information, it did not stand the same way to other forms of data that are sequential. To be more specific, it did not appear as a best

approach for performing textual corpus. So, *RNN* is another class of *deep neural networks* which process variable length sequence of inputs or sequential data as in textual corpus, this makes it more relevant towards *NLP* since a textual corpus is made of sequence of words and therefore as sentences.

Jurafsky and Martin (2019) mentions that *RNN* is any network that contains a cycle within its network connections where the value of a unit is directly, or indirectly, dependent on earlier outputs as an input. As sequential data is commonly divided by time, *RNN* accept inputs that correlate with data at a given time step. So, it also consists of a feedback loop where every time step's output is fed back to the network. This helps to access the record of the previous state to affect the output of future steps.

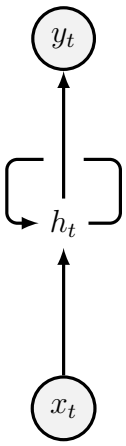


Figure 4.5: Simple recurrent neural network (Taken from Jurafsky and Martin 2019).

Figure 4.5 shows the structure of a simple *RNN* where an input vector representing the current input element x_t is multiplied by a weight matrix and then passed through an activation function to compute an activation value for a layer of hidden units which is later used to compute a output y_t. The recurrent link in the figure after passing the input augments the input to the computation at the hidden layer and is repeated as many times as necessary to cover all the time steps to process the whole sequence. This makes it look like we are performing the standard *feedforward* calculation, but the difference here is, we are also using a recurrence method which connects the hidden layer from the previous time step to the current hidden layer to make use of past context in calculating the output for the current input step.

Long Short Term memory

LSTM is a type of *RNN* which divides the context management problem into two sub-problems: removing information no longer needed from the context, and adding information likely to be needed for later decision making instead of using whole sequence (Jurafsky and Martin 2019). *LSTM*s accomplishes this problem by adding a context layer

to the architecture and by introducing the concept of gates to control the flow of information. Each gate in the network consists of a *feedforward layer*, and a sigmoid activation function to make the output similar to that of a binary mask.

While we are talking about *gates*, *LSTM* makes use of three types of gates: *forget gate*, *add gate*, and *output gate* (Jurafsky and Martin 2019). *Forget gate*, as the name implies is used to delete information from the context that is no longer needed and computes the weighted sum of the previous state's hidden layer and the current input and passes that through the activation function. Next is the *add gate* which is used to generate the mask from the result of the activation function to select the information to add to the current context and pass it to the *output gate* which then decides the information required for the current state.

4.4 Natural Language Processing

Text mining is the extraction of the relevant knowledge and patterns as meaningful information from the text data. Text mining is concerned with the detection of patterns in natural language texts (Popowich 2005). When accessing the textual information, applications can also benefit from a more detailed linguistic analysis of the text, as opposed to a shallower "word-based" analysis. As all the texts are certainly based on some language (*English, German, Italian, ...*), there arises the need of *NLP* which now is an important component of text mining.

NLP is a technique which deals with extracting the information from text data by learning different statistical techniques, and also involves the processing of words. And, during this process, it makes use of the words and its syntactical or semantical meanings, representation of entities. It also tries to establish the contextual representation of the word from the sentence or within sentences. *NLP* has been an active research topic producing new results, which also resulted in the development of various language processing models.

Among all those models, there is a recent discovery of pre-trained language model known as *BERT* which has proven to be the best in different learning tasks like, *question answering*, *natural language inference*, *classification*. The advantage of using *BERT* is that it can easily grab the context of the content because of its bi-directional language model. Using *BERT* model, it is also easier to fine-tune for various range of target tasks which justifies the application of *NLP*.

In this paper, we utilize various procedures and algorithms like *Bag of words (BoW), TF-IDF* for text mining that were before *BERT*, and also introduce *LM* which is the base

of our main implementation model.

4.4.1 Word Embeddings

Word embeddings are the numerical representations of words, usually in the shape of a vector (Mandelbaum and Shalev 2016). Words in a sentence or a document can have multiple occurrences. So, to capture that, each word is represented with a binary value (0 or 1), also known as *one-hot encoding* to represent the presence of the word in the sentence. This, as a whole, only tells us whether the word is available in the document or not. They are unsupervised learned word representation vectors whose relative similarities correlate with semantic similarity. Mikolov et al. (2013) mentions that distributed representations of words in a vector space help learning algorithms to achieve better performance in *NLP* tasks by grouping similar words. Mikolov et al. (2013) also introduced the Skip-gram model for learning high-quality vector representations of words from a large amount of unstructured text data.

> Let us take an example of two sentences, and see how the words are represented as discussed above:
>
> **Sentence A**: King is to queen as man is to woman.
>
> **Sentence B**: He bowed to the queen.

A suitable approach to represent these words in a quantifiable way is to encode the features with a vector and every word's feature identifies its location in a binary way which is known as the *one-hot-encoded* vector. The resulting vocabulary from the two sentences would be a collection of all unique words from it which are [King, is, to, queen, as, man, woman, he, bowed, the]. Now, we would apply *one-hot encoding* to see which words from the sentences are available in the vocabulary which would result as:

Table 4.1: Word one-hot encoded table.

	King	**is**	**to**	**queen**	**as**	**man**	**woman**	**he**	**bowed**	**the**
Sentence A	1	1	1	1	1	1	1	0	0	0
Sentence B	0	0	1	1	0	0	0	1	1	1

Looking at the Table 4.1, we only see the availability of word in each sentence. But, we cannot derive the representations like the relation between the words *king*, *man*, *queen*, and *woman*. This type of representation is also known as *BoW* representation which is discussed in Section 4.4.2 and is further used to create *word-count* vector with either containing 0 or 1 depending on the presence of the word in the vocabulary. If we are dealing with huge amount of training data, it might lead to a *sparse matrix* as most of the elements in the vector would be 0. Hence, this representation does not contain any meaningful relation and does not consider the order of words also meaning it does not capture the semantic relationship.

To achieve that, we would be requiring a word-vector representation, also known as *word-embeddings* which would keep the words having similar meanings closer to each other. Once the words are encoded, a subsequent vector space model (VSM) is modeled. This implementation was introduced by Mikolov et al. (2013) as *word2vec* which provides an efficient method for learning high-quality vector representations of words from large amounts of unstructured text data. And, it allows us to use vector arithmetics to work with word analogies like **king - man + woman = queen** as the word relationships can exist as linear substructures in the embedding space. Mikolov et al. (2013) also utilizes *distributional hypothesis*, which refers to the fact that the words are characterized by the words they hang out with also referring to as the context of the sentence. This also refers to the fact that the word "king" is more likely to be seen around the word "queen", and the word "man" is more likely to be seen around the word "woman" which is also presented in the Figure 4.6.

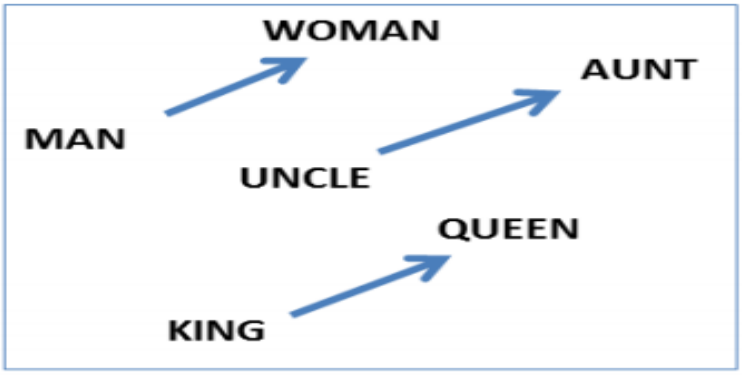


Figure 4.6: Word Embedding. (Mikolov et al. 2013)

This word analogy is further supported by Chen, Peterson, and Griffiths (2017) by introducing the parallelogram model of analogy which completes the word analogy *king : queen :: man : ?* by adding the difference vector between king and queen to man as represented in the Figure 4.7

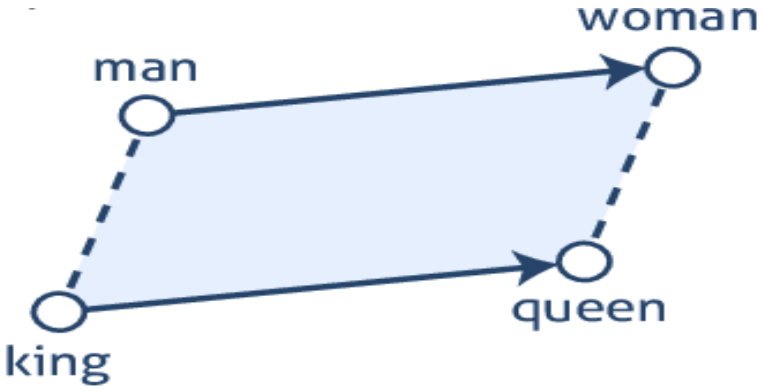


Figure 4.7: Parallelogram model of word analogy. (Chen, Peterson, and Griffiths 2017)

Word2Vec is a distributed word vector model that encodes words into embeddings using two different language modeling methods: the *continuous bag-of-words (CBOW) or the*

skipgrm models. *CBOW* works by predicting the probability of a target word within a window of given size, and *Skipgram* tries to predict the context word given the target word. While the *word embeddings* have several limitations like the size of the corpus, the order and independence of words, the major limitation of the vector approach is the inability to represent *polysemy*. *Polysemy* refers to the word having several meanings for it. This is due to the association of a single representation per word. To represent the *polysemy* situation, *Word2Vec* is unable to accurately learn its semantic or syntactic nature when we say "a river **bank**" or "a **bank** holiday". Here, the word *bank* is these two cases are different but *Word2Vec* builds a single representation for it depending on which came first.

4.4.2 Bag of Words (BoW)

The bag-of-words model is one of the most popular representation methods for object categorization (Zhang, Jin, and Z.-H. Zhou 2010). *BoW* is directly related to textual data, and is important for performing text classification. K and Joseph (2014) mentions that the *BoW* model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text is represented as an unordered collection of its words, disregarding grammar and even word order.

BoW model is only concerned whether known words occur in the document regardless of the position. It is the most common feature extraction process where the model represents text documents as vectors of some identifiers. The vector values are the frequency of the words in the document. *BoW* also ignores the context and meaning while discarding the order of the word in the document.

konkret	konservierung	kontext	kontrolliert	korrekte	krankheiten	kälber	kälberdurchfall	können	könnensomit
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	2	0
0	0	1	0	1	0	0	0	1	0
1	1	1	2	0	0	0	0	3	1
0	0	0	0	0	1	2	3	1	0

Figure 4.8: BoW representation example (NTS dataset).

Figure 4.8 shows the example of *BoW* representation of the documents from *NTS-ICD-10* dataset. Here we chose only 5 documents from the *NTS-ICD-10* dataset to illustrate the concept as an example which can be found at Appendix 1. Each column represents the word from the dataset, and each row represent a document from *NTS-ICD-10* dataset. As described earlier, each cell value represents the word frequency which makes each row as a feature vector for the document.

4.4.3 Term Frequency - Inverse Document Frequency (TF-IDF)

Term Frequency (TF) is a measure to count the occurence of a term/word in a document, whereas *Inverse Document Frequency (IDF)* is a measure to see the level of information a word provides, i.e. whether common or rare occurence. Then, the *IDF* value is logarithmically scaled to highly prioritize the term that has rare occurence and not to give full priority to the word with high occurrence.

TF-IDF calculates values for each word in a document through an inverse proportion of the frequency of the word in a particular document to the percentage of the documents the word appears in (Ramos 2003). Words with high Tf-IDF numbers imply a strong relationship with the document they appear in, suggesting that if that word were to appear in a query, the document could be of interest. TF-IDF is a numerical statistics that shows the relevance of keywords to some specific documents or it can be said that, it provides those keywords, using which some specific documents can be identified or categorized (Ramos 2003).

Thus, *TF-IDF* increases the importance of rare words and decreases the importance of more frequent words relative to the whole dataset. *TF-IDF* can be expressed using the following formulas:

$$w_{\mathrm{i,j}} = tf_{\mathrm{i,j}} \log\left(\frac{N}{df_{\mathrm{i}}}\right) \tag{4.5}$$

Equation 4.5 shows the calculation of *TF-IDF* utilizing the *TF* and *Document Frequency (DF)* where $tf_{i,j}$ is the frequency of term i within document j and *IDF* is calculated as the logarithmic inverse of *DF* with respect to N: the total number of documents. From Equation 4.5, we can also see that, *TF-IDF* is just the multiplication of *TF* and *IDF*. *BoW* plays a strong role by presenting a baseline representation for *TF-IDF* by not throwing out words and considering each word from the document which might be important for the direct representation or prediction.

konkret	konservierung	kontext	kontrolliert	korrekte	krankheiten	kälber	kälberdurchfall	können	könnensomit
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.039	0.000
0.000	0.000	0.039	0.000	0.048	0.000	0.000	0.000	0.027	0.000
0.048	0.048	0.039	0.097	0.000	0.000	0.000	0.000	0.082	0.048
0.000	0.000	0.000	0.000	0.000	0.043	0.086	0.130	0.024	0.000

Figure 4.9: Tf-IDF representation example (NTS dataset).

From the Figures 4.8 and 4.9, we can see the representation of the *BoW* and *TF-IDF*. If a *TF* for a word is very high in a document, and if it appears frequently in other documents, then its importance is scaled down using *IDF*. And, similarly if a *TF* for a words is very

high in a document, but it does not appear in other documents, then this term will have a greater *IDF*. This word becomes an important feature that helps to distinguish the particular document from other documents.

More specifically, in Figure 4.8, we take the words *konkret* which has occurence 1, and only occurs in 1 document, *kontrolliert* which has occurence 2, and only occurs in 1 document, and *können* which occurs in 4 documents with values 2, 1, 3, and 1. So, if we see the tf-idf values in Figure 4.9, the word *konkret* has 0.048 which is high as compared to *können* score (0.027 and 0.024) where its occurence is also 1 in a document. And, another word *kontrolliert* has occurence of 2 in a single document but do not occur in any other, so its *TF-IDF* score is 0.097 which is also high as compared to *können* score (0.082) where its occurence is 3 in a document. If we see in the figure, the word *können* is less prioritized as compared to others because it is mostly available in all documents.

This strongly supports the statement that, if *TF* for a word (*here, können, Figure 4.8*) is very high in a document and also appears frequently in other documents, then its importance is scaled down using *TF-IDF* as in Figure 4.9. But, the word *kälberdurchfall* which has a higher frequency and only appears in one document, it has a greater *TF-IDF* to increase its importance as in Figure 4.9.

4.4.4 Language Modeling

LM being the core component of *NLP* provides word representation and probability indication of word sequences (Jing and Xu 2019). A *LM* thus builds the characteristics of a language to see how the words relate to each other, and how likely is it for a word to occur given a sequence of words. Also, in some larger models, it also takes either the sentences or the whole paragraphs to build the context and use it to predict the further occurrences of words or sentences.

Here we will go through one of the basic type of *LM* known as *N-Gram models*. We also have *unigram* and *neural network language model*, but we will only see *N-Gram models* which was an approximation method and was most widely used state-of-the-art model before *neural network language models* (Jing and Xu 2019). So, *N* in *N-Grams* generally represent a number, where, if $N = 1$, it is known as *unigram model*, and if $N = 2$, it is known as *bigram model*, and so on. For $N = 1$, each word is taken as a unit and its probability is calculated by counting its occurence in the document and divided by the total amount of words. And, it goes similarly for all *N-Gram models* by taking into account all the previous *word n-grams*.

N-gram LMs have a drawback like assigning a probability of 0 to the *n-gram* that do not occur in the training corpus, and it cannot well handle modeling on large textual

corpora (Jing and Xu 2019). So, this *curse of dimensionality* problem led to the development of *neural network language model* where *deep neural networks* like *RNN* are known to automatically learn features and representations.

4.4.5 NLP Tasks

NLP has effectively been able to perform different tasks like *part-of-speech (POS)* tagging, *text classification, meaning*, *dialog systems*, *named entity recognition* and many more. While performing complex *NLP* tasks, it is common to be broken down into multiple sub-tasks to achieve the desired goal. These nlp tasks is also known as *downstream tasks* these days which represents only the *target tasks*.

One of the popular task is *text categorization* which groups various sub-tasks like *sentiment analysis, document classification, spam analysis*, and more classification tasks. They make use of some *pre-trained* models which already have a trained vocabulary, and later these tasks are only trained with the *target tasks* which makes it faster. And, another popular task is the *named entity recognition* which falls under *word sequence tasks* and is mostly relevant for language modeling as the derived task include the prediction of the previous and next words. And, we also have *word-embeddings* as mentioned in Section 4.4.1 which are useful for tasks requiring *text meaning* such as *question answering, search* and others.

NLP tasks are not just limited to the one mentioned above, but it has a wide range of tasks that can effectively use *natural languages* and run inferences on it to get some meaningful information out of it. It is applicable in all domains like *science, medicine, legal, politics*, and more.

4.5 BERT

BERT is a language model based on attention and transformers. There are two main *BERT* models: $BERT_{base}$ and $bert_{large}$. $BERT_{base}$ model's architecture is based on *12* bi-directional layers, *768* hidden dimension layers, and *110M* parameters while $BERT_{large}$ is based on *24* bi-directional layers, *1024* hidden dimension layers, and *340M* parameters, and is heavily based on bi-directional self-attention mechanism as mentioned in Devlin et al. (2019). Another important characteristics of *BERT* that makes it different from *ELMo* is the bi-directionality where *ELMo* is only based on traditional left-to-right and right-to-left trained *LM* which are later concatenated.

In this section, we will discuss about the concept of *Attention* and *Transformers* that are essential for the understanding of *BERT's* architecture. The general overview of *BERT* with its input structure, some details on pre-training methods and fine-tuning of

the model are discussed in Section 2.2.

4.5.1 Attention

Neural machine translation is a recently proposed approach using an encoder-decoder model to perform machine translation which aims at building a single neural network that can be jointly tuned to maximize the translation performance or for solving sequence-to-sequence (Sutskever, Vinyals, and Le 2014) problems. Despite their flexibility and power, Sutskever, Vinyals, and Le (2014) mentions that it can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality. The fixed size of this vector makes this method ineffective when dealing with longer sequences since it can't retain all the information and tends to forget the initial inputs. This issue might make it difficult for the neural network to better deal with long sentences, and hence deteriorate the performance of the encoder-decoder model. To overcome this problem, attention mechanism was introduced which adds a layer between the encoder and decoder to capture global information from the input sequence.

Bahdanau, Cho, and Bengio (2016) proposes an approach as an extension to the encoder-decoder model which learns to align and translate jointly which does not attempt to encode a whole input sentence into a single fixed-length vector, but it encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding the translation. Given a task of English-to-French translation, the proposed approach achieves, with a single model, a translation performance comparable, or close, to the conventional phrase-based system. For this task, the author uses a bidirectional RNN as an encoder and a decoder that emulates searching through a source sentence, and an alignment function that assigns the score for each pair of words. The function used in this case is a non-linear tanh activation function.

The task above describes a common scenario of how attention can be used to improve the sequence-to-sequence task such as machine translation. However, the attention mechanism is not just limited to the alignment function discussed above, but there are various types of attention mechanisms which we will be discussing in our next section like self-attention, scaled dot product attention, multi-head self-attention and are relevant to the transformer architecture.

Self-Attention

Self-attention, also known as intra-attention is an attention mechanism relating to different positions of a single sequence to compute a representation of the sequence (Vaswani et al. 2017). This mechanism is well known to be used successfully in a variety of tasks like *reading comprehension, summarization, textual entailment*, and more. To build rep-

resentations, each word of the input sequence is paired with the previous words with the highest attention score. This mechanism is useful in handling disambiguation and has a strong impact on *NLP* tasks like *machine translation* or *summarization*.

4.5.2 Transformers

Transformer is a neural network architecture proposed by Vaswani et al. (2017) where it implements the encoder-decoder model. Besides that, it relies on multi-head self-attention mechanisms in each encoder and decoder block. As mentioned about the translation example in Section 4.5.1, this mechanism is highly performant for solving *NLP* oriented problems like language translation and syntactic parsing.

As seen in Figure 4.10, we have a transformer architecture proposed by Vaswani et al. (2017), which on the left side is a *encoder model*, and on the right side is a *decoder model*. The inputs as we see are input embeddings for encoder and output embeddings for decoder. The *N* represents the number of blocks in both model, and in this context, both encoder and decoder are composed on 6 blocks. In this section, we will discuss about the *encoder model*, *decoder model*, *positional encoding*, and *multi-head self-attention* respectively.

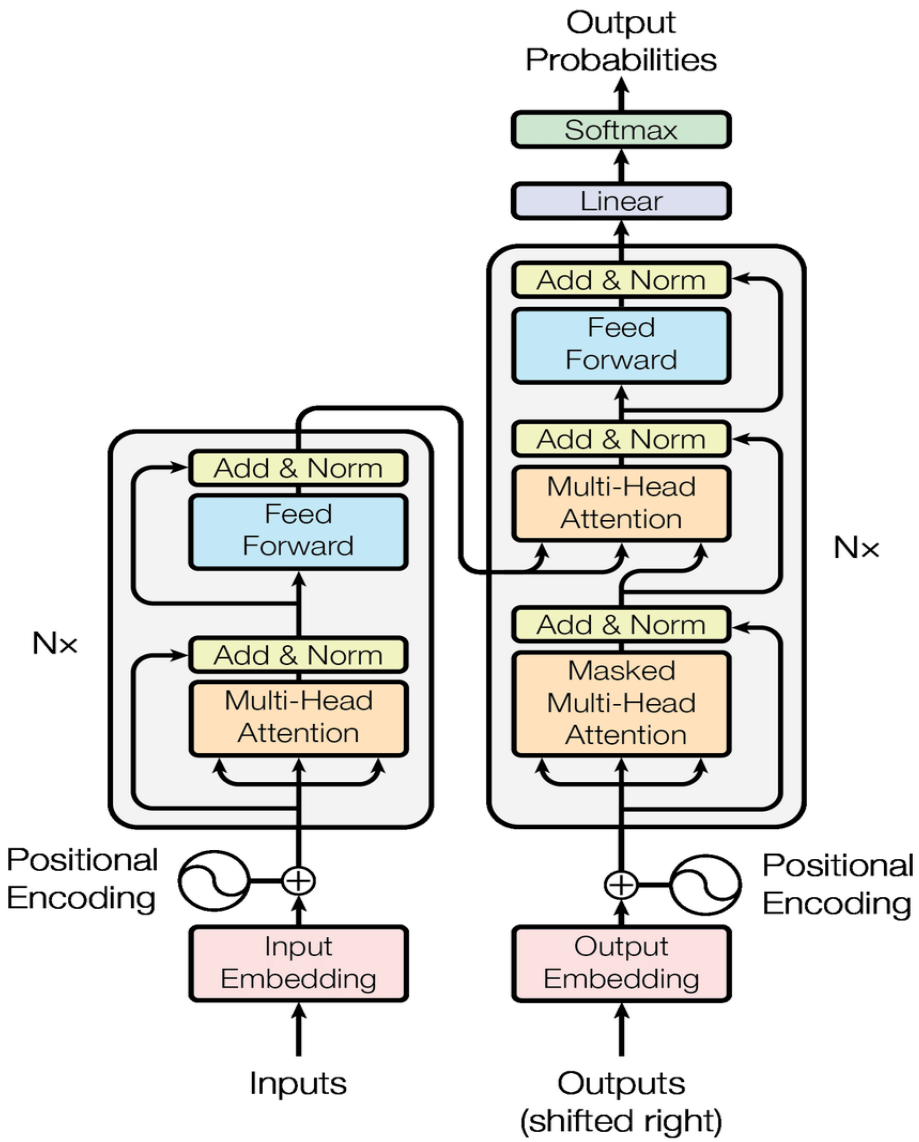


Figure 4.10: Transformer Architecture (Taken from Vaswani et al. 2017).

Encoder

The encoder model as mentioned in Section 4.5.2 consists of two main blocks (which will later include the sub-blocks inside encoder) as seen in Figure 4.11.

Out of the two main blocks of encoder model, first main block is the *Multi-Head* attention, and the other is the *Feed forward* network. Between both layers here, there is layer normalization present which adds the input to the output of a sub-network. However, the calculations here are efficient as it performs parallelized multiplications, and is the reason behind the rise of *transformers*.

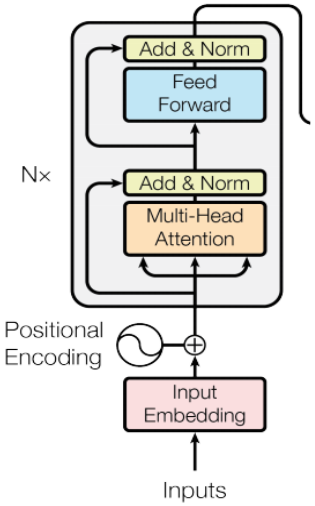


Figure 4.11: Encoder Model (Taken from Vaswani et al. 2017).

Decoder

The decoder model in Figure 4.12 is very similar to encoder, but has one additional layer than encoder which is *Masked Multi-Head Attention* which is used to attend over the previous decoder states. The decoder model masks the inputs to the decoder from future time steps. To make the decoder only consider the words occurring before the target word during translation tasks, decoders generally mask the future tokens when decoding a certain word. This makes sure that the prediction is only based on the words before the current word.

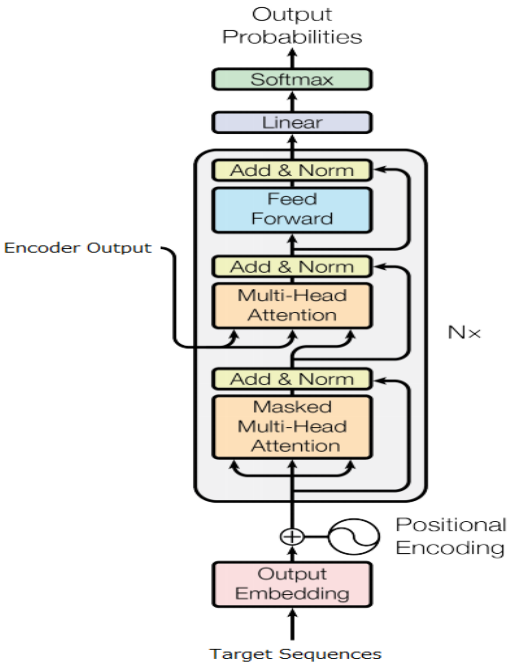


Figure 4.12: Decoder Model (Taken from Vaswani et al. 2017).

Positional Encoding

The translation task is highly efficient only if the tokens are placed in the correct position during or after prediction. So, both encoder and decoder model in Figure 4.10 have positional encoding combined to the input embeddings at the bottom of the stack as a multi-head attention network cannot naturally make use of the position of words in the sequence. Positional encodings explicitly encode the position of the inputs as vectors and are equal in dimension as input embeddings which makes it possible for adding both. Vaswani et al. (2017) mentions about two types of ways for calculating positional encodings: learned and fixed, the authors chose learned as the fixed performed just as well.

Multi-Head Self-Attention

As seen in Section 4.5.2, the encoder and decoder layer processes input and output embeddings into the attention sub layer known as *multi-head self-attention*. The attention mechanism, thus in the *transformer* is interpreted as a way of computing the relevance of a set of **values** based on some **keys** and **queries**. This mechanism is used as a way for the model to focus on relevant information.

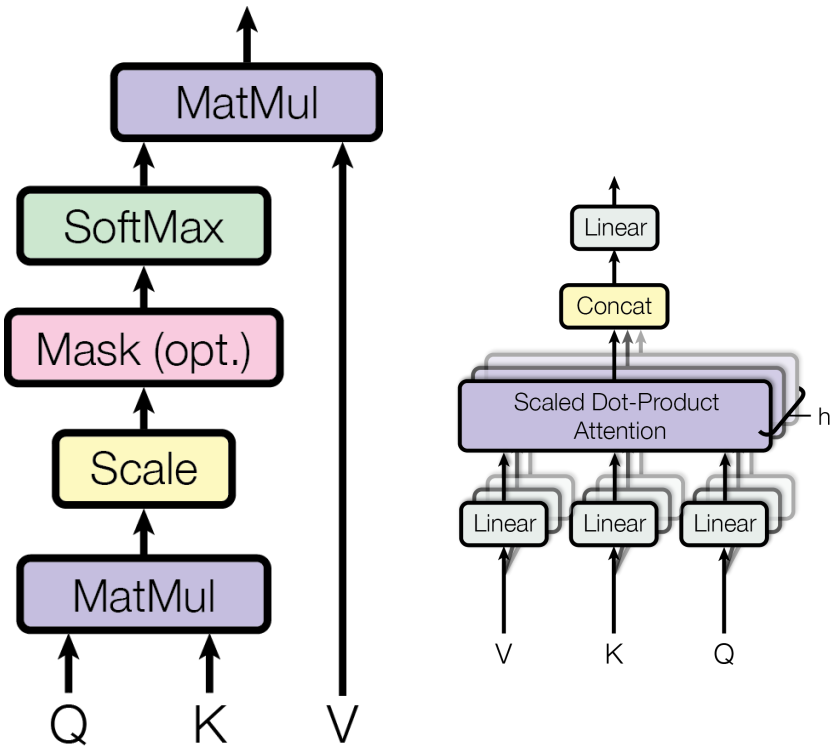
Before going deeper into *multi-head attention*, we also have *scaled dot-product attention* which is the base for *multi-head attention*. *Scaled dot-product attention* computes the attention on a set of queries, simultaneously packed together into a matrix Q, and the keys and values are also packed together into matrices K and V (Vaswani et al. 2017). So, the attention function is written as follows:

$$\text{Attention(Q, K, V)} = V \cdot \text{softmax}\left(\frac{QK_{\text{T}}}{\sqrt{d_{\text{k}}}}\right) \tag{4.6}$$

As seen in Equation 4.6, Q, K, and V are query, key, and value respectively, and d_k represents the dimension of the queries and keys. The softmax function here is used as a normalization function that transforms the components of the resulting vector into a probability distribution. The scaling factor $\frac{1}{\sqrt{d_{\text{k}}}}$ is used to scale the dot product as the matrix multiplication for larger values can require more time and space.

The multi-head attention mechanism utilizes the scaled dot-product attention multiple times in parallel which are then concatenated and linearly transformed into the expected dimensions. This approach then allows the model to jointly attend to information from different representation subspaces at different positions (Vaswani et al. 2017).

Figure 4.13 shows about the processes we described above about *scaled dot-product attention* and *multi-head attention*. If we go back to Figure 4.10, we can see that the output of attention is normalized and passed to the feed-forward network in both encoder and



(a) Scaled Dot-Product Attention (b) Multi-Head Attention

Figure 4.13: (Left) Scaled Dot-Product Attention, (Right) Multi-Head Attention. (Taken from Vaswani et al. 2017)

decoder layers. And, the output of the encoder block is fed to the attention layer in decoder where this layer is also preceded by a *masked multi-head attention* that processes the output embeddings to make sure that the predictions are based on known outputs for previous positions.

4.6 Domain Adaptation

While the definition of domain differs in each field of *ML*, so domain adaptation is a class of *transfer learning* where the scenarios could be:

1. Adaptation between different corpora.
2. Adaptation from a general dataset to a specific dataset.
3. Adaptation between subtopic in the same corpus.
4. Cross-lingual adaptation.

This thesis focuses on the adaptation of a model trained with general language to the medical language, the domain adaptation thus corresponds to the second category listed above which is an adaptation from a general dataset to a specific dataset. The textual data from a specific domain is more likely to display a particular vocabulary belonging to the given field, but could also attribute different semantics to words that are common to general domain language. Also, using the vocabulary already present in the *general*

dataset trained model, meaningful word embeddings can be produced for the texts in the *specific dataset*. While this is one way of adapting the vocabulary, another method is to produce a separate vocabulary for the specific dataset which would be more effective because there would be no *out-of-vocabulary (OOV)* words. But here, we adapt to the vocabulary present in the *general dataset* and observe the performance.

4.7 Feature Extraction

BERT has bi-directionally trained word-embedding which can also be called contextualized word embeddings which can be extracted and applied to custom *target task NLP* models in a similar way to how other word-embeddings can be used. This feature is known as *feature-based* which we discussed in Section 2.2.2 and are also similar to that of *ELMo*. The embeddings are generally in the lowest layers which can better handle the syntax and semantics of the words, and as it grows towards the top layer, the model has already learned more representations like attention and context. Hence, the original *BERT* paper (Devlin et al. 2019) mentions that the best results were found by concatenating the top four hidden layers of the model.

4.8 Multi-label Classification

Classification is the problem of assigning a document with one or more labels. Concerning this, there are three different types of classification problems. If a document is assigned with only one label out of two labels, it is known as a binary classification problem. In multiclass classification, there are more than two classes, and a document is assigned with only one class from it. In a multi-label classification problem, each document can be assigned with one or more labels.

Multi-label classification is the task of assigning an object simultaneously to one or multiple classes (Ghamrawi and McCallum 2005). A common way to perform multi-label classification is with a binary classifier for each class. However, the binary relevance method for multi-label classification was not so adequate to its label interdependencies and introduced a novel chaining method that can model label correlations while maintaining acceptable computational complexity (Read et al. 2009). This chaining method passes label information between classifiers, allowing the classifier chain to take into account label correlations and thus overcoming the label independence problem of binary method.

The documents from the dataset *NTS-ICD10* used in this paper have one or more labels assigned to it. Therefore, we treat it as a multi-label text classification problem. For example, one document can be assigned with multiple diagnoses like (**Tumorwachstum, Eierstockkrebs, Brustkrebs**).

4.9 Evaluation

As discussed by Yang (1998), micro and macro performance are inspired by the performance evaluation in multiclass text classification where classes are often not distributed equally. As in our case, classes represents the labels (ground truth labels) that are already available in the presented *NTS-ICD-10* dataset. Using these evaluation measures allows us to compare performance both for documents that are more or less linked to the frequent classes.

Micro average, or per-document average aggregates the occurrence or contribution of all classes and then average them to compute the metric. On the other hand, macro average performs on the class level (here per-label level) giving equal weight to each ground truth labels, regardless of its frequency which means that it computes the metric independently for each class and then take the average. The aim of using these performance measures for evaluation is to avoid misinterpretation of results when some dominant labels are always predicted correctly and labels with few examples are not in case of *class-imbalance*.

According to Asch (2013), *micro averaging* gives equal weight to each per-document classification, whereas *macro averaging* gives equal weight to each class. With respect to the *precision* and *recall*, the *micro averaged* results are more effective measure to see its impact on the large classes while *macro averaged* results are more effective on small classes which we will seen in detail in example Section 4.9.5. Both micro and macro average performance measures use the performance indicators *Precision (P)*, *Recall (R)*, and *F1-score (F1)*. These indicators are further computed over *True Positive (TP)*, *False Positive (FP)*, and *False Negative (FN)* assignments. And, the assignments for *TP*, *FP*, and *FN* would be better represented in a *confusion matrix* which are later useful in calculating the *P, R*, *F1* scores.

For the computation as mentioned in Pilz (2015), we replace classes with ground truth labels and define micro and macro performance as follow. Let l^*(d) be the ground truth label for a document *d* and l^+(d) be the predicted label. If the prediction of a model is correct, i.e. l^+(d) = l^*(d), we have a *TP*:

$$tp(l^*(d)) = \begin{cases} 1, & \text{if } l^*(d) = l^+(d) \\ 0, & \text{else.} \end{cases} \tag{4.7}$$

If the prediction is not correct, i.e. $l^{+}(d) \neq l^{*}$(d), we have a *FN* for l^{*}(d):

$$fn(l^{*}(d)) = \begin{cases} 1, & \text{if } l^{*}(d) \neq l^{+}(d) \\ 0, & \text{else.} \end{cases} \tag{4.8}$$

Similarly, we have a *FP* for l^{+}(d):

$$fp(l^{*}(d)) = \begin{cases} 1, & \text{if } l^{*}(d) \neq l^{+}(d) \\ 0, & \text{else.} \end{cases} \tag{4.9}$$

Now, to compute the micro and macro average performance measures, let us assume a collection of documents $\mathbf{D} = (d_i)_{i=1}^{|D|}$ with associated ground truth labels $\mathbf{L_D} = l^{*}(d_i)_{i=1}^{|D|}$ and the further calculations are shown in sections 4.9.2 and further[2]. As we saw the representation for *TP*, *FP*, and *FN*, we will further discuss the *confusion matrix* in Section 4.9.1 and *P, R, F1* in Sections 4.9.2, 4.9.3, and 4.9.4 respectively. And, after this we will also work on an example that discusses about *micro* and *macro* averaging scores in Section 4.9.5.

4.9.1 Confusion Matrix

Confusion matrix is a useful way for showing precision and recall metric given the prediction labels from a model. It shows four different outcomes: *TP*, *FP*, *True Negative (TN)*, *FN* placed in a table as a matrix form. The actual values are represented as columns and the predicted values are represented as rows.

Table 4.2: Confusion matrix table in case of multi-class or multi-label classification

		Actual			
		A	B	C	D
Predicted	A	TP	FP	FP	FP
	B	FP	TP	FP	FP
	C	FP	FP	TP	FP
	D	FP	FP	FP	TP

Table 4.2 shows the sample confusion matrix in the case of *multi-class* or *multi-label* classification. The four main outcomes from the confusion matrix are represented as:

- **True Positive (TP):** documents predicted as positive that are actually positive.
- **False Positive (FP):** documents predicted as positive that are actually negative.

[2]Symbols https://www.researchgate.net/figure/Ground-truth-square-symbols-vs-predictions-plus-symbols-for-each-period-of_fig14_333675748

- **True Negative (TN):** documents predicted as negative that are actually positive.
- **False Negative (FN):** documents predicted as negative that are actually negative.

All the *TP* are the items in the diagonal of the matrix. The columns represents the *actual labels* while the rows represent the *predicted labels*. While calculating the *precision* and *recall* metric, we are only concerned with *TP, FP*, and *FN*. So, all the row items except the diagonal item is known as *FP*, and all the column items except the diagonal item is known as *FN* for the specific row or column label.

4.9.2 Precision

In information retrieval and classification problems, precision is the metric that tells us how many documents were actually correct from the documents retrieved. Precision is the number of true positives divided by the sum of true positives and false positives.

Micro average metric will aggregate the contributions of all classes to compute average metric, while macro average metric will compute metric independently for each class and then take the average.

Micro average metric will first compute the total number of *TP*, *FP*, and *FN* over all documents:

$$TP = \sum_{d_i \in \mathbf{D}} tp(l^*(d_i)), FP = \sum_{d_i \in \mathbf{D}} fp(l^*(d_i)), FN = \sum_{d_i \in \mathbf{D}} fn(l^*(d_i)) \tag{4.10}$$

Then, micro *P* is computed as:

$$P_{micro} = \frac{TP}{TP + FP} \tag{4.11}$$

In opposition to micro average metric, macro average metric first computes $P(P_{macro})$ for each label *l* in the ground truth set $\mathbf{L_D}$:

$$P_{macro}(l) = \sum_{d_i \in \mathbf{D}} \frac{tp(l^*(d_i))}{tp(l^*(d_i)) + fp(l^*(d_i))} \tag{4.12}$$

where, $l^*(d_i)$ = *l*;
Then, this value is averaged over all ground truth labels $\mathbf{L_D}$:

$$P_{macro} = \frac{1}{|\mathbf{L_D}|} \sum_{l \in \mathbf{L}_D} P_{macro}(l) \tag{4.13}$$

4.9.3 Recall

Recall is the metric that tells us how many documents from labelled set were identified as a match from the retrieved set. Recall helps to find all the relevant documents from the dataset. Recall is the number of true postives divided by the sum of true positives and false negatives. Similar to precision, we will be evaluating both micro and macro averaged metric for recall.

With support of Equation 4.10, let us calculate micro *R* as:

$$R_{micro} = \frac{TP}{TP + FN} \tag{4.14}$$

Similar to Equation 4.12, we will calculate $R(R_{macro})$ for each label *l* in the ground truth set $\mathbf{L_D}$ as:

$$R_{macro}(l) = \sum_{d_i \in \mathbf{D}} \frac{tp(l^*(d_i))}{tp(l^*(d_i)) + fn(l^*(d_i))} \tag{4.15}$$

where, $l^*(d_i)$ = *l*;
Then, this value is averaged over all ground truth labels $\mathbf{L_D}$:

$$R_{macro} = \frac{1}{|\mathbf{L_D}|} \sum_{l \in \mathbf{L}_D} R_{macro}(l) \tag{4.16}$$

where $|\mathbf{L_D}|$ is the number of distinct labels in $\mathbf{L_D}$.

4.9.4 F1-score

F1-score is a metric which helps to combine both precision and recall score and get a balanced score. The F1-score is the harmonic mean of P and R which takes both metric into account. F1 reaches its best value at 1 and worst at 0 letting us measure the test's accuracy.

With support of Equation 4.11 and 4.14, let us calculate micro $F1(F_{micro})$ as:

$$F1_{micro} = \frac{2 * P_{micro} * R_{micro}}{P_{micro} + R_{micro}} \tag{4.17}$$

And, similarly using Equation 4.13 and 4.16, let us calculate macro $F1(F_{macro})$ as:

$$F1_{macro} = \frac{2 * P_{macro} * R_{macro}}{P_{macro} + R_{macro}} \tag{4.18}$$

Both $F1_{micro}$ and $F1_{macro}$ are computed over the ground truth labels $\mathbf{L_D}$ and not over all

possible predictions, which are all existing labels in the NTS-ICD dataset. The recall for labels not available in the ground truth collection $\mathbf{L_D}$ cannot be computed in a meaningful way as there are no document containing those labels. So, if a model prediction $l^+(d)$ is not available in the ground truth $\mathbf{L_D}$, this is counted as a false negative for the ground truth target $l^*(d)$ but not as a false positive for the predicted entity $l^+(d)$.

4.9.5 Example: Micro and Macro Performance

In this example, we will see the *micro* and *macro* averaging performance in case of *multi-label* classification. We will see two scenarios: *when all the labels are correctly predicted* and *when one label gets mislabelled*.

Consider a collection of documents D = {d1, d2, d3, d4} with associated ground truth labels $\mathbf{L_D}$ = $\{l^*(d_1) = \{l_1, l_2\}, l^*(d_2) = \{l_1, l_3\}, l^*(d_3) = \{l_1, l_4\}, l^*(d_4) = \{l_2, l_4\}\}$. Further from this assignment, we have four set of ground truth labels $\mathbf{L} = \{l_1, l_2, l_3, l_4\}$. If all the documents are linked correctly to their respective labels, then referring to the confusion matrix Table 4.2, it would be constructed as:

Table 4.3: Confusion matrix table when all labels from each documents are linked correctly to the actual ones

		Actual			
		l_1	l_2	l_3	l_4
Predicted	l_1	3	0	0	0
	l_2	0	2	0	0
	l_3	0	0	1	0
	l_4	0	0	0	2

Now, referring to the confusion matrix 4.3, we can see the *TP, FP*, and *FN* as:

$$tp(l_1) = 3, tp(l_2) = 2, tp(l_3) = 1, tp(l_4) = 2$$
$$fp(l_1) = fp(l_2) = fp(l_3) = fp(l_4) = 0$$
$$fn(l_1) = fn(l_2) = fn(l_3) = fn(l_4) = 0$$

This results in micro and macro performance values of:

$$P_{micro} = R_{micro} = F1_{micro} = 1$$
and
$$P_{macro} = R_{macro} = F1_{macro} = 1$$

In this case, it would give a perfect score of 100% which shows the best-case scenario. But, in the case where one label from a document d_2 is mislabelled, which means a prediction error of l_3 to l_2, we would have the following confusion matrix:

Table 4.4: Confusion matrix table when one label from a document is mislabelled with another label (Label l_3 from document d_2 is mislabelled with l_2)

		Actual			
		l_1	l_2	l_3	l_4
Predicted	l_1	3	0	0	0
	l_2	0	2	1	0
	l_3	0	0	0	0
	l_4	0	0	0	2

Now, referring to the confusion matrix 4.4, we will first calculate the *P* for all the labels and later relate it with the average scores:

$$tp(l_1) = 3, tp(l_2) = 2, tp(l_3) = 0, tp(l_4) = 2$$
$$fp(l_1) = fp(l_3) = fp(l_4) = 0, fp(l_2) = 1$$
$$fn(l_1) = fn(l_2) = fn(l_4) = 0, fn(l_3) = 1$$
$$p(l_1) = 1, p(l_2) = 0.66, p(l_3) = 0, p(l_4) = 1$$

f According to Equations 4.10, 4.11, 4.14, and 4.17, the micro performance results would be:

$$P_{micro} = R_{micro} = 0.875, F1_{micro} = 0.875$$

and, similarly, following Equations 4.12, 4.13, 4.15, 4.16, and 4.18, the macro performance results would be:

$$P_{macro} = 0.666, R_{macro} = 0.75, F1_{macro} = 0.705$$

As discussed earlier, micro performance gives equal weight to each document samples, the high number of correctly interpreted labels has more impact and we noticeably obtain a higher micro performance. Whereas, macro performance states that only half of the ground truth labels were correctly retrieved, i.e. label l_3 was misinterpreted and predicted as l_2. The low macro performance here could imply that some documents are more difficult to link to their actual labels or there is a huge imbalance in the ground truth labels which has a negative impact on the model performance which can be further verified by looking at the precision scores of l_1 and l_3. Label l_1 has the highest occurence in the samples which has all been correctly retrieved resulting in a $p(l_1)$ of 1, thus the higher score on the *micro averaged* score, whereas label l_3 has only occurred once which has been mislabelled resulting in a $p(l_3)$ of 0, thus the lower score on the *macro averaged* score.

This example smoothly explains the approach of *micro* and *macro* averaging where *micro* gives more priority to the domination class, so the *micro* score (0.875) is near to the *P* of the dominating class which is 1. Similarly, *macro* gives equal weight to all class which explains that its score (0.705) is more inclined towards the *P* of the imbalanced class l_3 which is 0.

Chapter 5

Experiments

In this chapter we will dive into the experiments we conducted to achieve the main goal of this thesis. In Section 5.1, we will see a brief overview of the various experiments we conducted, and the modules we used for it. After that, in Section 5.2 we conducted the fine-tuning of German *BERT* using domain-specific datasets (medical domain) as seen in Section 3.4 and observed the training and loss performance on various dataset combinations. This section will help us determine to choose the best-fine-tuned model to later perform the token masking experiment and classification task. This is also the part that will lead us to get the answer to the research questions crafted in Section 1.3. So, moving in that direction, Section 5.3 performs the experiment of tokenization masking and prediction with several *BERT* baseline models and *fine-tuned* model and help us answer the first research question. After that, in Section 5.4 we will see a brief overview on the implementation of classical models for the classification tasks using the evaluation dataset as seen in Figure 3.1, and we will see in detail for the several *BERT* models where we train these models for the classification tasks using the evaluation dataset. This is the section that will help us answer our second research question. Lastly in Section 5.5, we will see the performance metrics of all those models and also discuss about the masked token prediction and *ICD-10* code classification result by various *BERT* models and evaluate them.

5.1 Experimental Setup

In this section, we will be discussing the setup of the various experiments that we conduct in the further sections. We used the training dataset, dev dataset, and test dataset from the provided corpus from *NTS-ICD-10* animal experiments. We used the training dataset to train our multi-label models. We tried several baseline models like LinearSVC, LogisticRegression, and FastText and implemented those models to see their performance on the development set and test set accordingly. FastText can simply handle multilabel problems provided with its specific dataset format. But, for LinearSVC and LogisticRegression, it has to be wrapped with OneVsRestClassifier to handle it as a multi-label classification problem. We used all the baseline models and its supporting modules from the *sklearn*

library.

We built *TF-IDF* vectors as input text representations by only choosing top 20000 tokens as a vocabulary and MultiLabelBinarizer to represent multi-label model where we remove all the classes with a frequency less than 25. And, we did not add any placeholder label to support documents without any annotated *ICD-10* codes. As we will know all the *ICD-10* codes available beforehand, we initialized MultiLabelBinarizer with all the *ICD-10* codes or labels from the set of train datasets.

Apart from those classical baseline models, we also used multi-lingual *BERT* language model and German *BERT* language model as the main baseline models as the performance of these models would be more specific for us to compare with the performance of the fine-tuned model on the domain-specific dataset, which in case there is a medical domain. And, to utilize *BERT* models, we used the transformers library from hugging face with PyTorch implementation.

We have not performed any hyperparameter optimization, but chose the baseline parameters for traditional methods as suggested in Amin et al. (2019), and used the suggestions from Devlin et al. (2019) for the training of *BERT* models. We used the provided evaluation script from *sklearn* module and report *P*, *R*, and *micro F1*-score as evaluation metrics.

5.2 Fine-tuning German BERT model with medical domain dataset

During implementation of *BERT* models, we took *Multilingual BERT* and *German BERT* as baselines for the classification of *NTS-ICD* documents, but the evaluation is focused in the *German medical BERT* model which we obtained after fine-tuning the *German BERT* using the medical datasets that we collected from the internet. To fine-tune for this task, we used *Hugging Face* library to create an adaptive model with the classification prediction head. The hyperparameters that we applied are maximum sequence length of 512 tokens which is the maximum limit of *BERT*, 3 epochs, batch size of 8 along with gradient accumulation steps of 4, and learning rate of 5e-5. We only used 3 epochs as it was the max number of epoch to be used for *fine-tuning* as suggested in Devlin et al. (2019). Also, we had to use batch size of 8 and gradient accumulation steps of 4 which ultimately makes the model to understand that we are using a batch size of 32.

After evaluating the performance of the *German BERT* model, it is now time to perform *fine-tuning* on it using the *medical* datasets to see if there would be any improvement in the behavior. So, we prepared datasets for *fine-tuning* from Section 3.4 in two different ways:

1. Sentences (Dataset having one sentence per line.)
2. Articles (Dataset having token length upto *BERT* maximum token length.)

Here, after reaching the maximum token length after traversing several sentences from the *medical* datasets, we would start a new row in the dataset creation.

Also, we *fine-tuned* and *evaluated* on 4 different use-cases to see if the model trained on sentences is evaluated well on articles and vice-versa. The use-cases are listed below:

1. Sentence train - Sentence eval
2. Sentence train - Article eval
3. Article train - Article eval
4. Article train - Sentence eval

The purpose of running all these different types of usecases is to properly choose the *fine-tuned* model for our classification tasks for *NTS-ICD-10* datasets. We would choose the best model based on the eval loss from all these use cases. So, it would either be sentence trained or article trained *fine-tuned* model.

Observing the nature of our *NTS-ICD-10* classification documents, each document are multiple paragraphs long, which also imitates the behavior of an article. This might also be in our favour as *BERT* was also trained on *wikipedia* articles and, this being noted, the loss behavior for article would be better than sentence.

5.2.1 Usecase 1: Sentence train - Sentence eval

Figure 5.1: Sentence training using scraped medical articles - Sentence eval on NTS documents loss plot for fine-tuning.

Figure 5.1 shows us the loss plot for the *fine-tuning* usecase when the model was trained with additionally collected *medical* sentences and evaluated with *NTS* sentences from *NTS-ICD-10* dataset. This experiment consisted of 6,36,507 lines of medical texts as a training set collected from several medical sources listed in Section 3.4.

We ran this experiment altogether for 6 epochs with a batch size of 8. Instead of capturing the loss at every epochs, we did that for every 5000 batch steps, so the total loss points would be calculated as:

> Train samples: 636507
> Epoch: 6
> Bath size: 8
> Batch step: 5000
> Total training steps: (Train samples / Batch size) =
> : 79563.75
> : equivalent to 79564 steps.
> Total loss points: Epoch * (Total training steps / Batch step)
> : 6 * (15.91 (equivalent to 16))
> : 96 steps.
> Eval samples: 1000

Each epoch in this manner has 16 loss points, which can further also be called as loss point interval. The train loss in the plot above is stable after 40th loss point, and the minimum eval loss captured until this position is 2.1. This tells us that the model is already well *fine-tuned* at 3 epochs, and if we pick the loss point interval, it will be between 33 to 48 steps for third epoch. And, if we see at the minimum eval loss which does not decrease beyond that value, we capture it at the 33rd position, the value being 2.1.

This usecase and behavior also strongly supports the *BERT* paper by Devlin et al. (2019) which mentions that the effective number of epochs for *fine-tuning* is either 2 to 3.

5.2.2 Usecase 2: Sentence train - Article eval

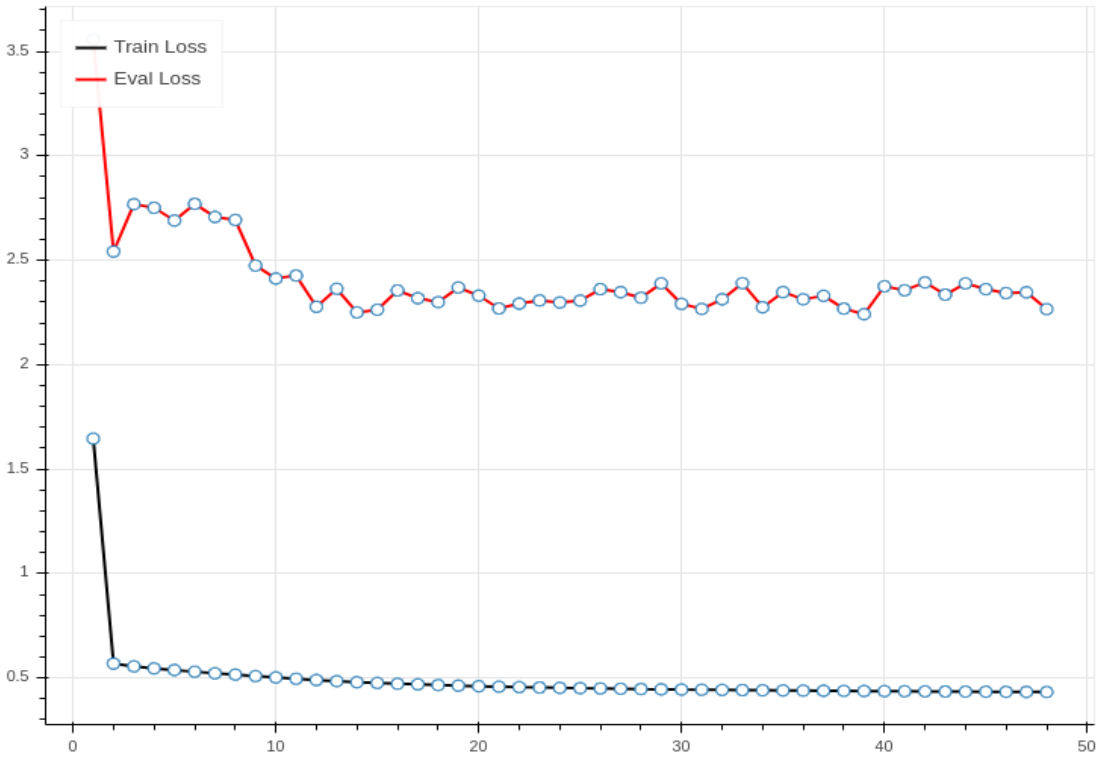


Figure 5.2: Sentence training using scraped medical articles - Article eval on NTS documents loss plot for fine-tuning.

Figure 5.2 shows us the loss plot for the *fine-tuning* usecase when the model was trained with additionally collected *medical* sentences and evaluated using documents from the *NTS-ICD-10* dataset. This experiment also consists of the same configurations as that in usecase 5.2.1 except that we train it for 3 epochs and only use 100 *NTS* documents as a evaluation sample.

Considering the loss point interval as discussed in usecase 5.2.1, the minimum evaluation loss of 2.25 is seen at 14^{th} step although the training loss is still decreasing. And, going more further we do not see any improvement in the evaluation loss which is not an achievement for *fine-tuning*. The minimum evaluation loss was captured in 14^{th} step which falls in first epoch, and this model's evaluation loss did not perform well as evaluated from usecase 5.2.1 and *BERT's* paper by Devlin et al. (2019) also in the case of utilizing all 3 epochs.

5.2.3 Usecase 3: Article train - Article eval

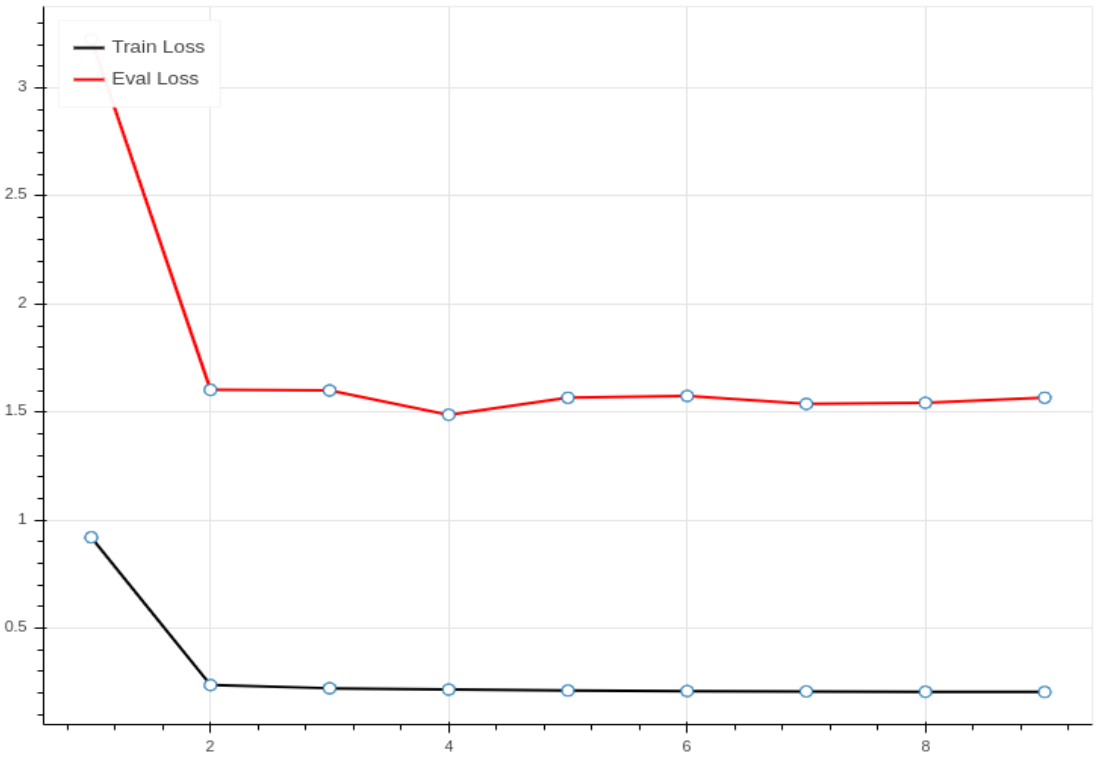


Figure 5.3: Article training using scraped medical articles - Article eval on NTS documents loss plot for fine-tuning.

Figure 5.3 shows us the loss plot for the *fine-tuning* usecase when the model was trained with additionally collected *medical* articles and evaluated using documents from the *NTS-ICD-10* dataset. This experiment consisted of 34,283 lines of medical articles as a training set collected from several medical sources listed in Section 3.4.

We ran this experiment altogether for 3 epochs with a batch size of 8. Instead of capturing the loss at every epochs, we did that for every 1500 batch steps, so the total loss points would be calculated as:

> Train samples: 34283
> Epoch: 3
> Bath size: 8
> Batch step: 1500
> Total training steps: (Train samples / Batch size) =
> : 22.85
> : equivalent to 23 steps.
> Total loss points: Epoch * (Total training steps / Batch step)
> : 3 * (2.875 (equivalent to 3))
> : 9 steps.
> Eval samples: 100

Each epoch in this manner has 3 loss points, which can further also be called as loss point

interval. The train loss in the plot above is stable after 2nd loss point, and the minimum eval loss captured is 1.5 but only in 4th interval. This tells us that the model is already well *fine-tuned* at 2 epochs.

This usecase and behavior also strongly supports the *BERT* paper by Devlin et al. (2019) which mentions that the effective number of epochs for *fine-tuning* is either 2 to 3. And, this might be our *fine-tuned* model as the *NTS-ICD-10* classification documents are also composed of articles, and the model is learning well as to get minimum loss on the evaluation dataset. But, we still have one more evaluation model to compare the loss before choosing the right model.

5.2.4 Usecase 4: Article train - Sentence eval

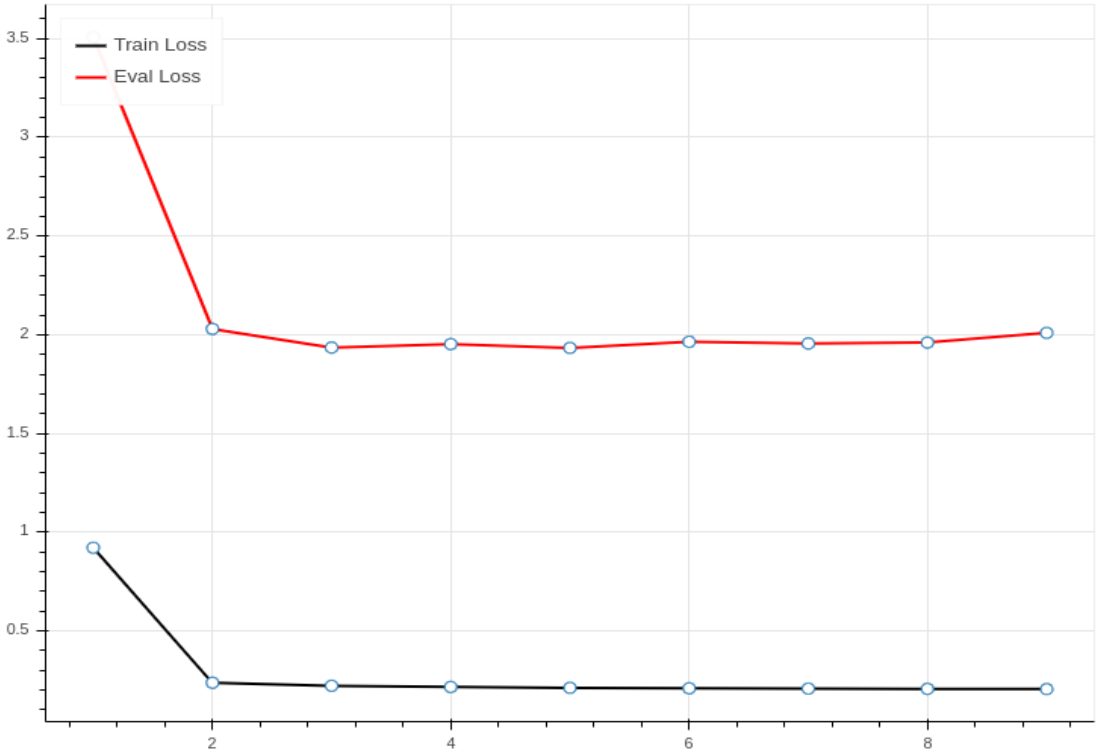


Figure 5.4: Article training using scraped medical articles - Sentence eval on NTS documents loss plot for fine-tuning.

Figure 5.4 shows us the loss plot for the *fine-tuning* usecase with similar configuration as in 5.2.3, but with 1000 evaluation samples.

Similar to the model in 5.2.3 with a evaluation loss of 1.5, the evaluation loss for this model is minimum at 3rd loss point with a value of 1.9. This model evaluates well for *nts-articles* as it was trained using *medical articles*, but the evaluation loss for *nts-sentences* is also not less compared to the *sentence* trained model. Apart from the evaluation loss, this *medical articles* trained model also achieves less train loss than compared to the model trained with *medical sentences*.

5.3 BERT Predicting Masked Tokens Experiment

We used the training objective known as *MLM* from *BERT* whose goal is to efficiently predict the masked token from a sequence of tokens. This is an effective way of checking the bi-directional conditioning of *BERT* model as the masked word has already been seen by the model's deep bi-directional model as discussed in Section 2.2. For doing this, *BERT* (Devlin et al. 2019) suggests to mask some percentage of the input tokens at random, and then predict those masked tokens.

As *BERT* was originally trained on *Wikipedia articles* and *books corpus*, and our dataset was related to *medical* terminologies, we were eager to see the the performance of *German BERT* model on this task for both *Wikipedia articles* and *medical articles* datasets and compare their performance as we believed it would be difficult for this model to really catch on the *medical* terms. *MLM* predictions can be done for two type of scenarios:

1. SubWord masking
2. WholeWord masking

When we tokenize a sequence or a sentence using *BERT* model, it replaces it with the nearest word-piece combination that it can find from the models[1] vocabulary. So, a sample tokenization for a sentence

> ER-Analysen bei neurodegenerativen Erkrankungen

would be

> ER, -, Analysen, bei, neur, ##ode, ##gener, ##ativen, Erkrankungen

Here, we can see that the word *neurodegenerativen* is replaced with the nearest word-piece combinations as this word could not be found in the model's vocabulary, and the word-piece following the initial word-piece for the target word would be prepended by ##. This type of tokenization is known as *WordPieceTokenization*, and is the original tokenization implementation of *BERT*.

And, following this implementation, to perform predictions for *SubWord masking*, we would randomly mask the tokens from the tokenized sentence as

> ER, -, Analysen, [MASK], neur, ##ode, [MASK], ##ativen, [MASK]

And similarly, as for our sentence above, we only have one word which was separated into several pieces, so, the masking using *WholeWord masking* would look like

[1] German BERT vocab, https://int-deepset-models-bert.s3.eu-central-1.amazonaws.com/pytorch/bert-base-german-cased-vocab.txt

ER, -, Analysen, bei, [MASK], [MASK], [MASK], [MASK], Erkrankungen

Here we replace all word-pieces that were separated by the tokenizer, and the goal is to predict all the pieces belonging to that single word. But, while doing this we should also keep in mind not to mask all of the word-pieces because of which the model cannot learn properly as it loses most of it contextual data and hence result in poor performance. Hence, through this experiment, we will try to find the answer to the first research question crafted in Section 1.3.

5.3.1 Experiment 1: SubWord Masking

As discussed earlier, we created two separate datasets from wikipedia and our *NTS-ICD* datasets consisting of 10,000 sentences and 500 articles. In this experiment, we are masking 15% of the tokens from each sentence as proposed by *BERT* (Devlin et al. 2019).

Figure 5.5 shows us the accuracy of *SubWord* masking for both *wikipedia* and *NTS-ICD* dataset which were arranged sentence wise and article wise. The reason behind doing this was to see which arrangement of the dataset would catch the context properly so as to gain performance for the *MLM*. As said in the beginning of this section that the *German BERT* model would not be able to catch *medical* terminologies properly as it was trained using *wikipedia* articles and *book corpus*, the Figure 5.5 tells us otherwise. In both *nts-sentences* and *nts-articles* section, the score of *masked tokens* is higher than with *wikipedia*.

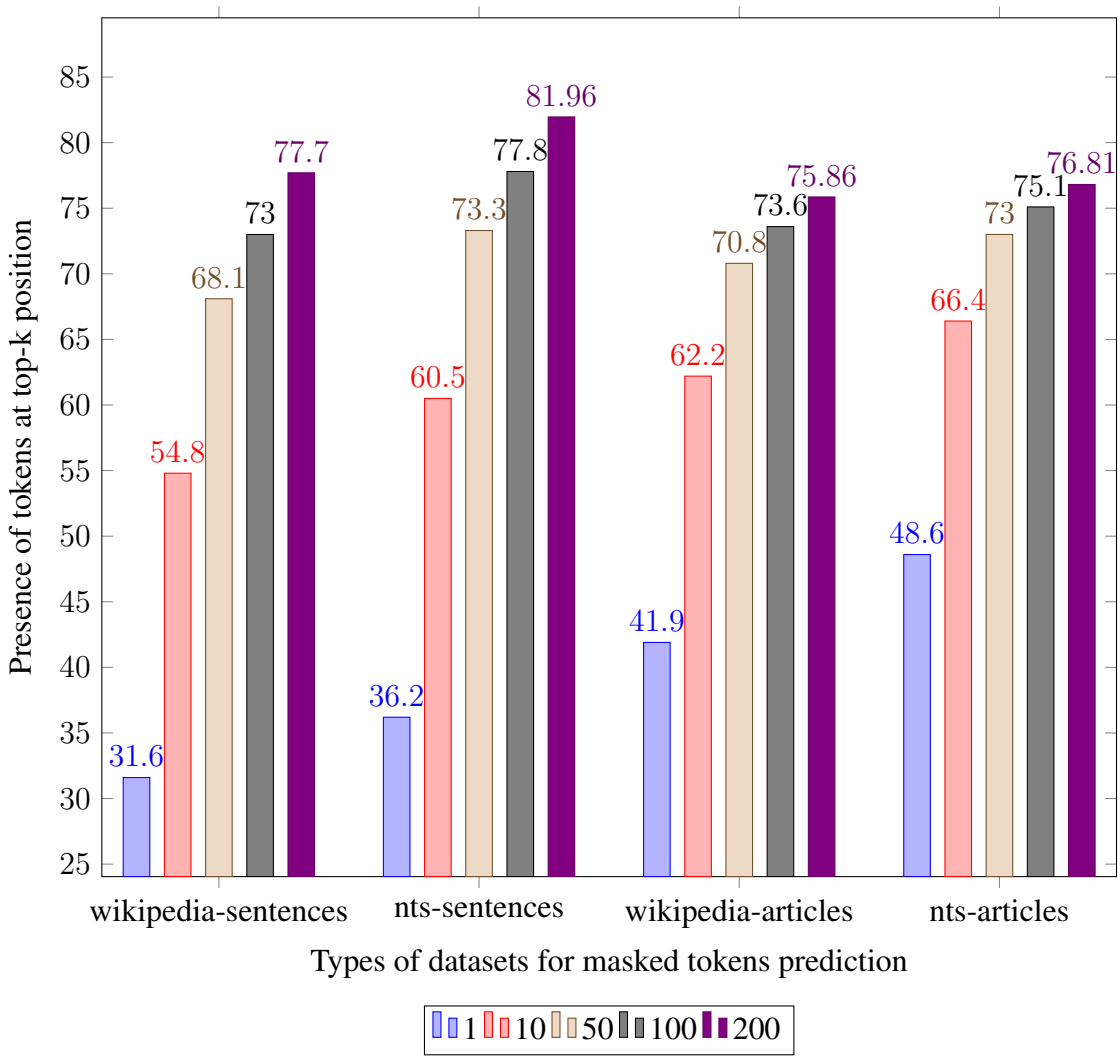


Figure 5.5: SubWord masking experiment and performance of German BERT model

Here will see a sample prediction for *sub-word* masking on *nts-sentences* using the German *BERT* model.

> **Sentence:** Die mult ##ipl ##e Sk ##ler ##ose (MS) ist eine aut ##oi ##mm ##une Erkrankung des zentralen Nerven ##systems .
> **Masked Sentence:** Die mult ##ipl ##e Sk ##ler [MASK] (MS) ist eine aut [MASK] ##mm ##une Erkrankung des [MASK] Nerven ##systems .
> **Masked tokens:** ['##ose', '##oi', 'zentralen']
> **Predicted tokens:** ['##ose', '##oi', 'zentralen']

The predicted tokens were collected within the *top-10* tokens from the model's prediction. We used *topk*[2] method with positions 1, 10, 50, 100, and 200 respectively to get the token predictions and see in which position was the correct token being predicted.

[2]topk https://pytorch.org/docs/master/generated/torch.topk.html

As we compare the scores between *wikipedia-sentences* and *nts-sentences* in the Figure 5.5, we see that the predictions for *nts-sentences* are already strong and better than *wikipedia-sentences* in the lowest positions and also maintains consistency in the higher positions. As we see the scores between the *wikipedia-articles* and *nts-articles*, the scores at positions 1, 10, 50 are comparably higher, but gradually decreasing as we evaluate it for more higher positions.

Hence, for the *sub-word* masking, the performance is better in *sentences* than in *articles* which could also be because of the reason that *BERT's* attention is much higher between the words in the same sentence rather than in multiple sentences which means that it can easily grab the context between the words in same sentence. Through this experiment, we are now convinced that this model can also grab *medical* contexts better than *wikipedia*, but it also shows us that this model requires some *fine-tuning* looking at the scores for *nts-articles*.

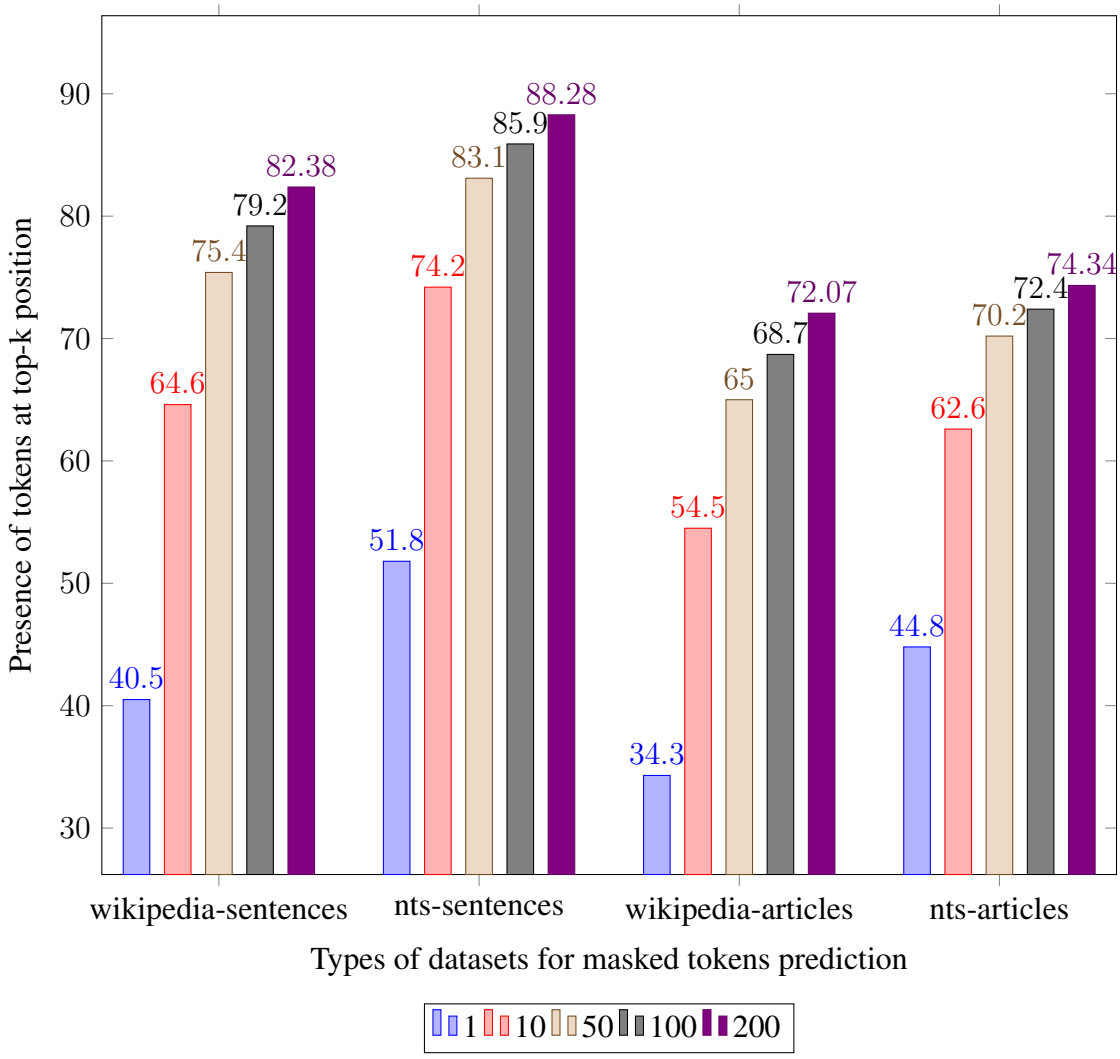


Figure 5.6: SubWord Masking experiment and performance of fine-tuned German BERT model with medical domain (Section 5.2.3)

Figure 5.6 shows the *sub-word* masking using the medical domain *fine-tuned* model. If we compare Figure 5.6 with Figure 5.5, the score for *nts-sentences* for *top-10* position has increased by 43% while for *nts-articles*, it has decreased by 0.078%. Looking at the scores, we can definitely say that *fine-tuning* did have some good effects on the *masked token* prediction performance. We will verify the result for sub-word masking for both models in Section 5.5.1 and compare them.

5.3.2 Experiment 2: WholeWord Masking

In this experiment, we are using the same datasets as used in Section 5.3.1. As we are doing *WholeWord masking* which is basically masking all word-piece tokens obtained after tokenizing the sentences, we would want to limit the number of *WholeWord* to be masked to prevent masking most of the words from the sentence and loose context which will result in poor performance. So, we will only mask two *WholeWords* from each sentences.

Figure 5.7 shows us the performance for *WholeWord masking* using the German *BERT* model. The result for the *nts* is worse than the *wikipedia* datasets which was not the exact result that we wanted to see. Although, if we compare the results between *wikipedia* and *nts*, there is a minimal difference between *sentences*. In the *articles*, *nts* score is higher than *wikipedia* in position one which is a good sign but, *wikipedia* has a gradual increment in rest of the positions which highly points out that this model needs some *fine-tuning* with domain related datasets. Comparing the scores in Figure 5.5 and Figure 5.7, the *German BERT* model is well capturing the context for *medical* documents as well in case of *SubWord masking*, but it fails to the same for *WholeWord masking*. So, as a next step, we *fine-tuned* the *German BERT* model with some *medical* documents.

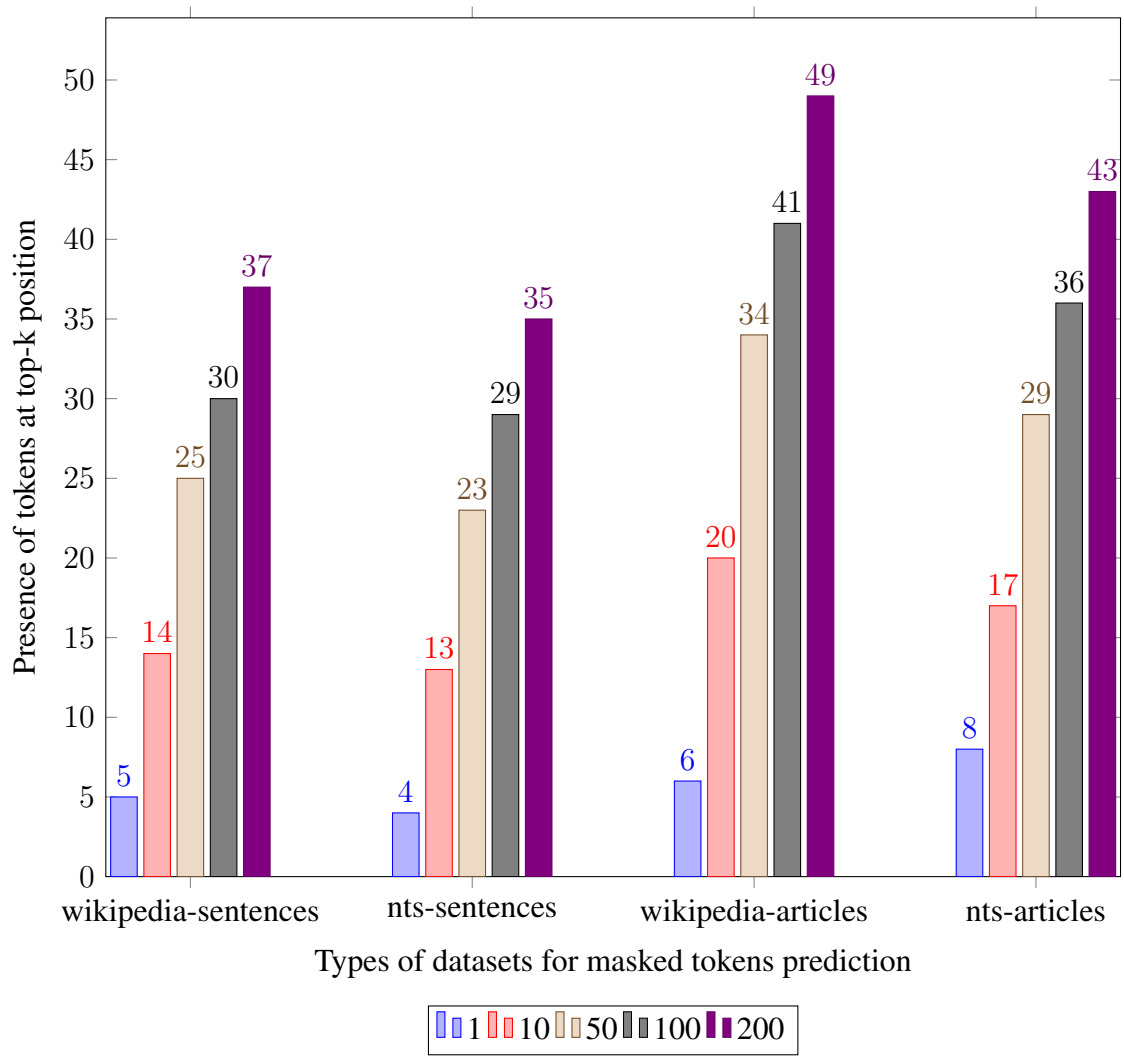


Figure 5.7: WholeWord Masking experiment and performance of German BERT model

Here, as done in Section 5.3.1, we will also see a sample prediction for *whole-word* masking on the same *nts-sentences*.

> **Sentence:** Die mult ##ipl ##e Sk ##ler ##ose (MS) ist eine aut ##oi ##mm ##une Erkrankung des zentralen Nerven ##systems .
> **Masked Sentence:** Die mult ##ipl ##e [MASK] [MASK] [MASK] (MS) ist eine aut ##oi ##mm ##une Erkrankung des zentralen Nerven ##systems .
> **Masked tokens:** ['Skl', '##er', '##ose']
> **Predicted tokens:** ['', '##er', '##ose']

We have the predictions which looks correct, but we also want to see the position it was found in the prediction set, as with *sub-word* masking, most of the tokens were already in the first position. This result is properly discussed in Table 5.5 of Section 5.5.1.

Figure 5.8 shows us the performance for Wholeword masking using the medical domain fine-tuned German *BERT* model. The result for both *nts-sentence* and *nts-articles* has gone better than the one in Figure 5.7. And, if we compare the results

between *nts* and *wikipedia*, there has been a improvement in the attention by the model for nts sentences and articles.

We can now verify the effects of *fine-tuned* model for *whole-word* masking as we did for *sub-word* masking in Table 5.3 and 5.4. The models performance improved by 46.15% on the *top-10* predicted tokens and by 28.57% on *top-100* predicted tokens for nts-sentences. This is a good sign that *fine-tuning* a model using *domain specific* datasets will improve the overall model performance.

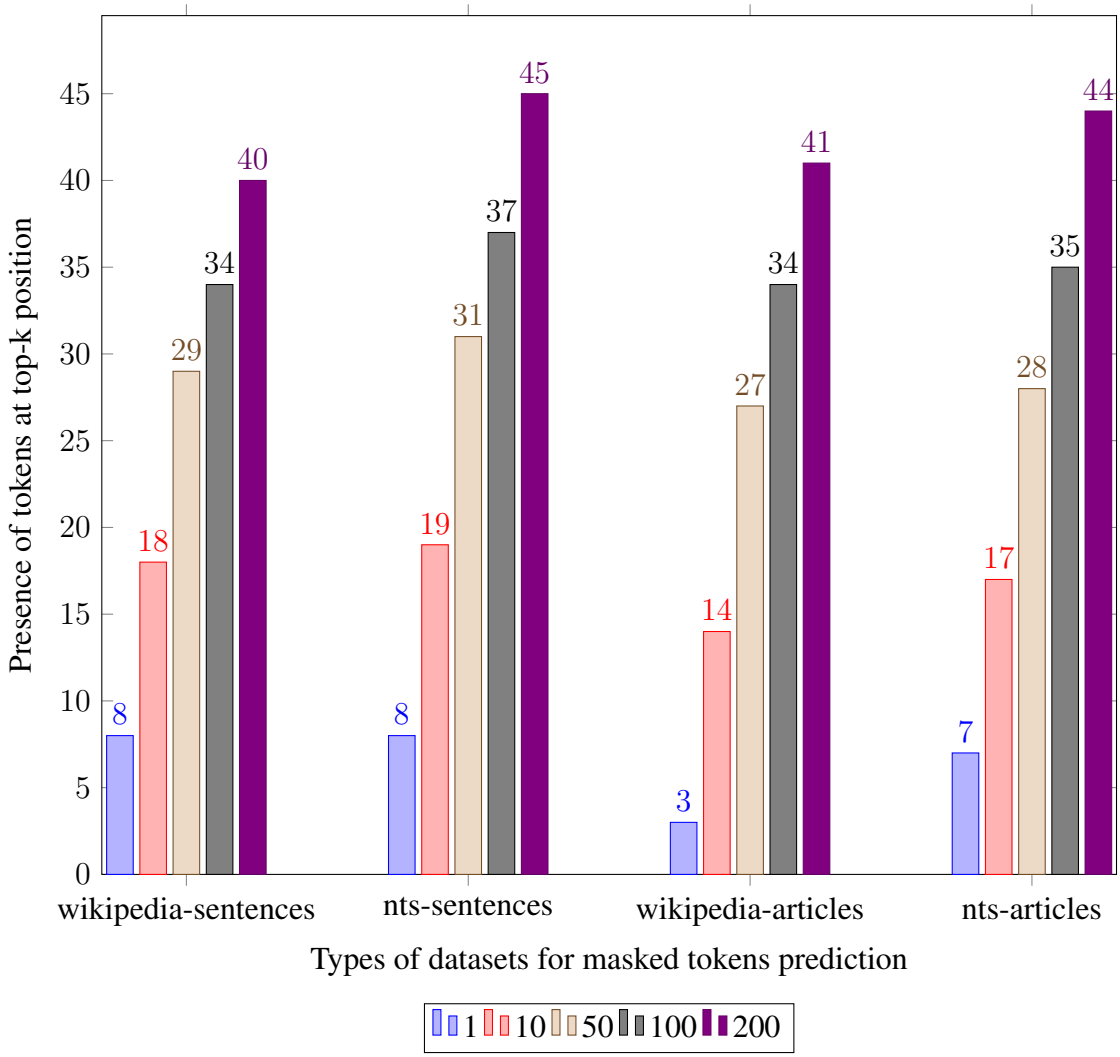


Figure 5.8: WholeWord Masking experiment and performance of fine-tuned German BERT model with medical domain (Section 5.2.3)

After having all these scores for the *MLM*, we also wanted to do a sanity check for the *MLM* scores for the sentences from *NTS* documents. So, we created a sample document of only ten sentences from *NTS* documents, and another document by randomly replacing the words from the *NTS* sentences with any word that does not fit the context to verify if the scores that we obtained are really the one that we can trust or not. The documents therefore can be found in Appendix 5.

As we can see in Figure 5.9, our score seems to be reliable as the *MLM* scores for original *NTS* sentences are better than *NTS* sentences filled with random words. This also means that the *BERT* is better in grabbing and understanding the contextual meanings given properly formed sentences so as to better predict the missing tokens.

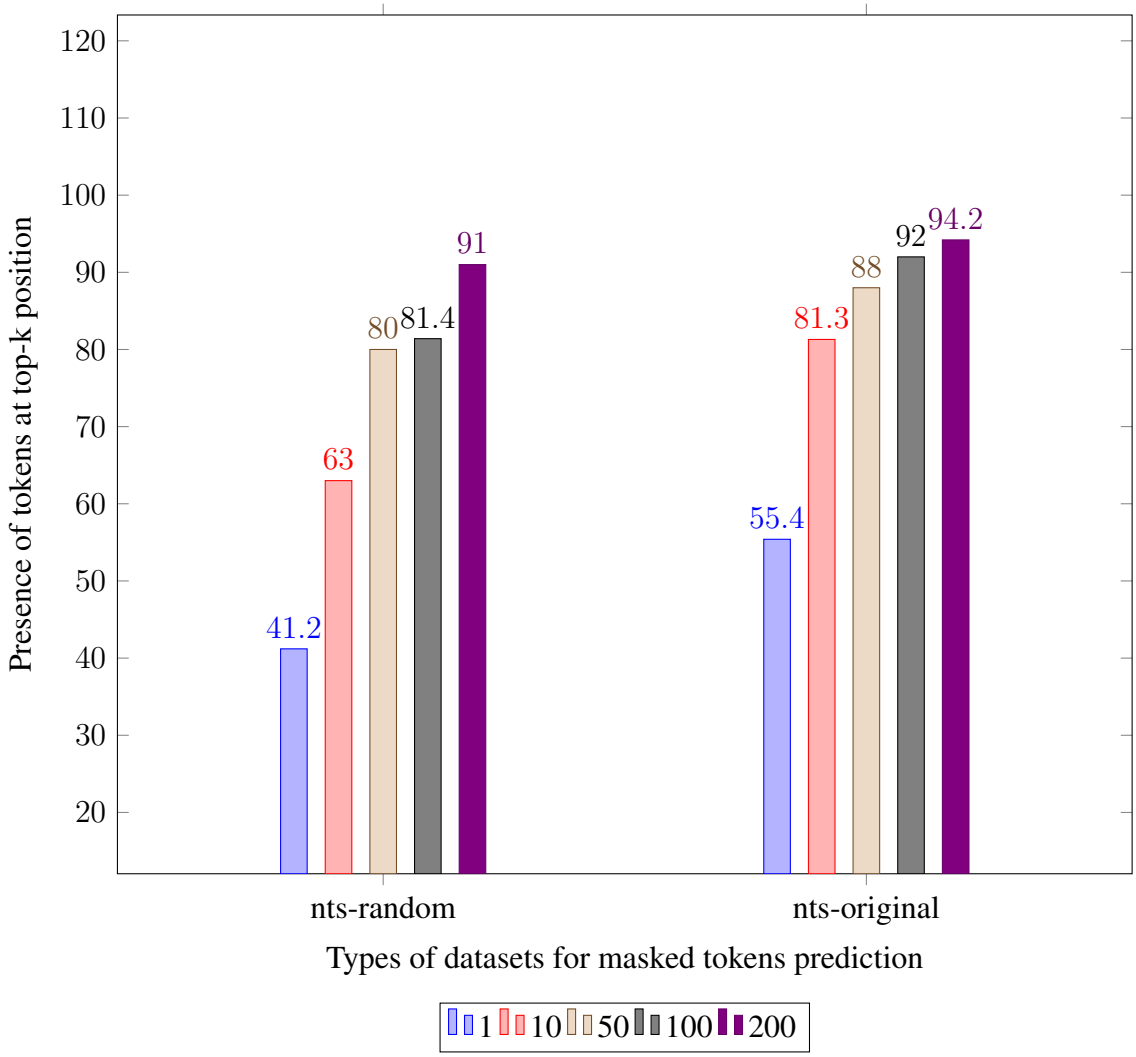


Figure 5.9: SubWord Masking experiment and performance of German BERT model when passing actual sentences(nts-original) and another as replacing words with some other words to break context(nts-random).

5.4 NTS-ICD-10 document classification

For the classifications tasks required for the *NTS-ICD-10* code classification, we compare the *BERT* model with other well-researched methods as baselines. Using this approach, we can compare the effectiveness of German *BERT* and our German medical *BERT* on the classification tasks mentioned above. Several baseline methods that we chose for this purpose are *LinearSVC, Logistic Regression, FastText, Multilingual BERT, German BERT*.

5.4.1 Linear SVC

Linear SVC is one of the most applicable machine learning algorithms which is used to predict the probability of a class to occur. In our case, we have implemented it to perform a multi-class classification problem where each class is assigned a probability between 0 and 1. And, similarly to make it useful for multilabel classification, we wrapped it with OneVsRestClassifier. It is often used as a baseline model because of its simplicity. To implement this, we used the Scikit-Learn (*sklearn*) library and the LinearSVC method along with OneVsRestClassifier. The relevant parameters were class weight set to balanced to automatically adjust weights inversely proportional to class frequencies, and max iter set to 5000 to let the model converge.

OnevsRestClassifier or One-vs.-rest strategy trains a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. Since each class from the problem is represented by single classifier only, it is possible to learn about the class by inspecting its respective classifier. Following this technique, it can be used for multilabel learning, where a classifier is used to predict multiple labels for an instance. [3] [4]

Linear SVC is similar to *Support Vector Classification (SVC)* having linear kernel. While *SVC* were developed for binary classification, it is now possible to perform multiclass and multilabel classification using OneVsRestClassifier. And, for a linear kernel or a model, there exists a hyperplane that completely separates the vectors into two non-overlapping classes. Using this separation, if perfect, it may result in a model with most of the cases that the model classifies correctly. But, if the separation is not perfect, then the result would be that the model classifies incorrectly for most of the cases.

5.4.2 Logistic Regression

Logistic regression is another applicable machine learning algorithm used to predict the probability of a class to occur. We implement this in the same way as we did it for Linear SVC to achieve the multiclass and multilabel classification problem. This is another baseline model that is equally simple to implement. To implement this, we used the Scikit-Learn (*sklearn*) library and the LogisticRegression method along with OneVsRestClassifier. The relevant parameters were class weight set to balanced, solver set to liblinear, and L2 normalization. Logistic regression is better used when the target variable is categorical, i.e., if we have to predict the class or labels by learning the representations from the textual documents.

[3] OneVsRestClassifier https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html

[4] One-vs.-rest strategy https://en.wikipedia.org/wiki/Multiclass_classification#One-vs.-rest

5.4.3 FastText

While several linear classifiers who often are better to build strong baselines and tend to be slow at training, Joulin et al. (2016) shows another linear model with a rank constraint and a fast loss approximation which can train on a billion words within ten minutes. Here, the word representations are averaged into a text representation, which is in turn fed to a linear classifier and uses the softmax function to compute the probability distribution over the predefined classes. During FastText implementation, we used the default classifier that comes along with Fast text library.[5] The hyperparameters used while implementing FastText are learning rate of 0.3, epochs of 50, bigrams, and loss of ova. *One-vs-All (OVA)* loss generally is helpful to us as it represents one-vs-all and works well with multi-label classification.

Joulin et al. (2016) uses hierarchical softmax to reduce the computational complexity when the number of classes is large and is also advantageous at test time when searching for the most likely class. Also, it makes use of n-gram features to capture partial information about the local word order.

Fast text is efficient to perform both supervised and unsupervised classification as well as efficient for learning word representations. It supports both *Continuous Bag of Words (CBOW)* and Skip-gram models training using negative sampling, softmax or hierarchical softmax loss functions. Fast text implementation is as easy as other linear model implementation, but it is scalable and fast as compared to others.

5.4.4 Bidirectional Encoder Representations from Transformers

In this section, we will discuss about the *NTS-ICD10* classification done using various *BERT* models like *multilingual bert*, *german bert*, and *medical fine-tuned bert*. As we treat this classification problem as a *multi-label* classification problem, we used *BCE* loss, as it easily handles this problem as a multiple binary classification problem. This solves the problem of class imbalance by giving equal attention or weight to all labels in the problem. All the classification tasks in this section were done using *Google Colab notebooks GPU*[6] which took around 5 to 6 hours depending upon the availability of the resources. The documents that we took for this sample prediction evaluation can be found in the Appendix 6.

[5] FastText https://fasttext.cc/docs/en/supervised-tutorial.html
[6] Google Colab https://colab.research.google.com

Classification 1: Multi-lingual BERT

Multilingual *BERT* (M-BERT), released by Devlin et al. (2019) is a single language model pre-trained from monolingual corpora in 104 languages with a 12 layer transformer (Pires, Schlinger, and Garrette 2019).

In this section, we also refer to the result of Amin et al. (2019) who used *multi-lingual BERT* to classify the *NTS-ICD10* problem, and achieved 76% as a *F1-micro* score on the german dataset. As a result of replicating the scores based on Amin et al. (2019), we have the following plot as seen in the Figure 5.10:

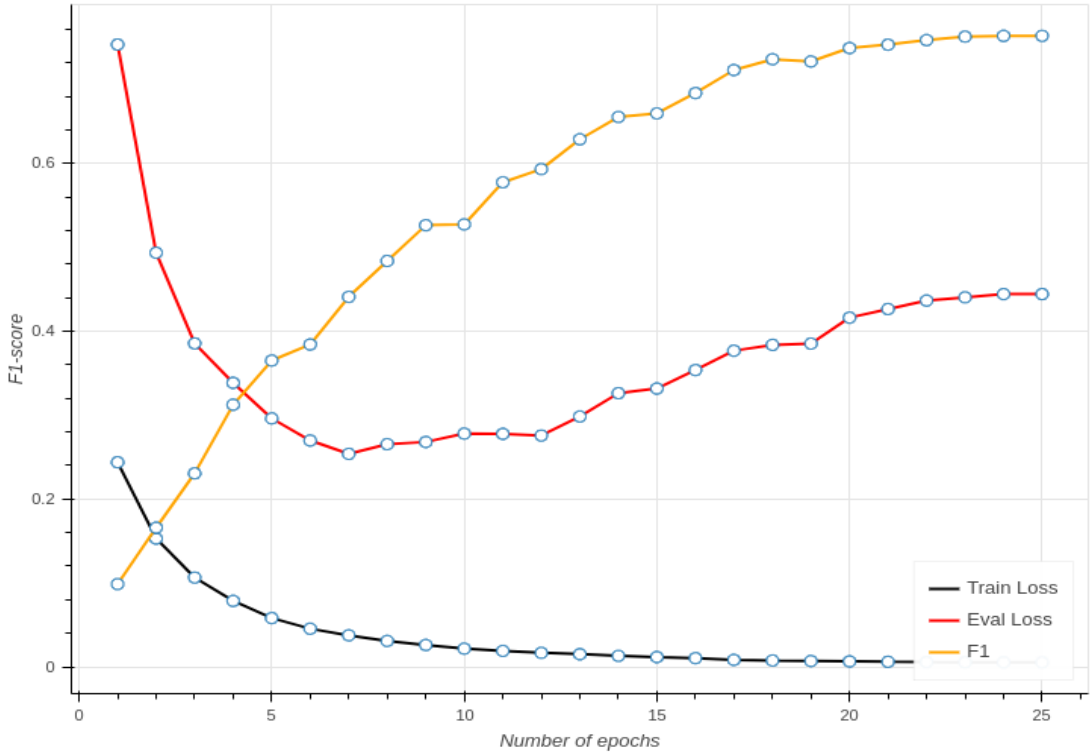


Figure 5.10: Multi-lingual BERT for *NTS-ICD-10* document classification (Amin et al. 2019)

The classification was done for 25 epochs as the *F1-micro* score was continuously improving with the *train loss* also doing the same. But, if we have a look in the curve of *eval loss*, the interpretation of the score goes otherwise. The *eval loss* in the Figure 5.10 is decreasing until epoch 7, and drastically starts increasing after it with no any signs of going down. The *F1-micro* score on dev set noted for this full run was 75.1%. This type of behavior indicates that the model is overfitting. As this happens, the model is training on the data too well, but fails to generalize to other data which in this case is the evalation dataset.

This behavior in the curve of *eval loss* could be because of reasons like, im-balanced dataset or with many unnecessary features, or also because of the loss method used. If it were the former two reasons, there would be some effect on the *train loss* too, so we tried changing the loss function used by Amin et al.

(2019), and we saw a improved curve for it which is presented in the Figure 5.11:

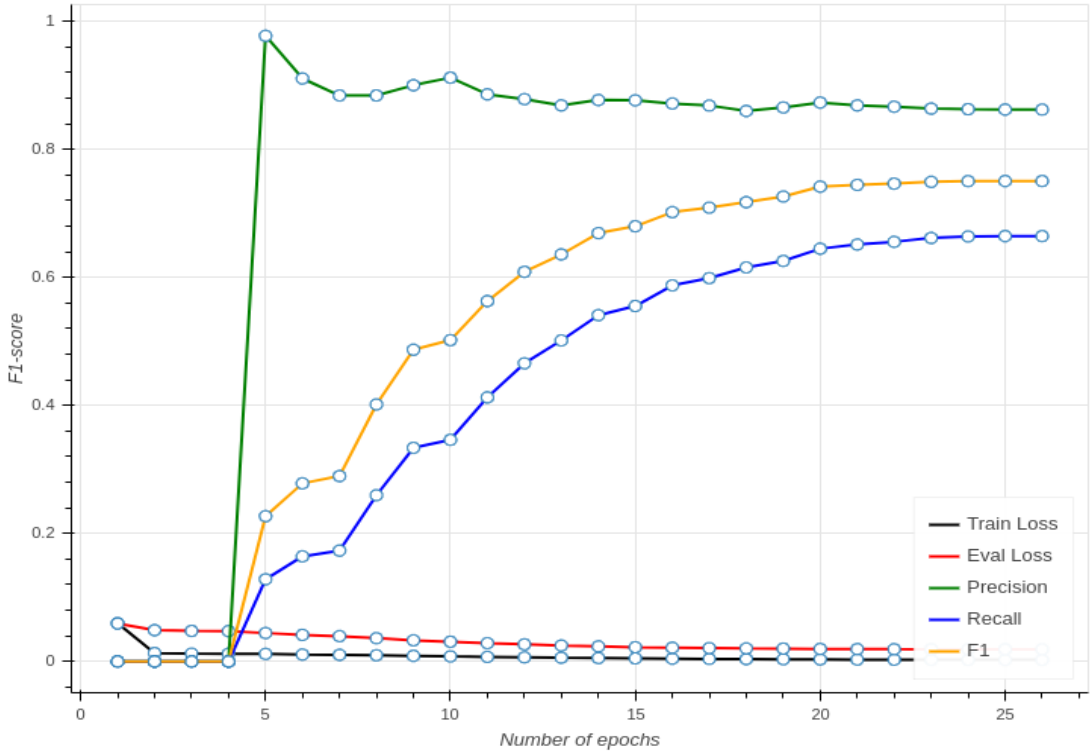


Figure 5.11: Multi-lingual BERT with *BCE* loss for *NTS-ICD10* document classification

For this implementation we used *BCE* classification[7] loss which is suitable for *multi-label*. And, as we see in the Figure 5.11, both *train loss* and *eval loss* are decreasing together in the same manner while the *F1-micro* score keeps on increasing. The *eval loss* stops improving along with *F1-micro* score after 24th epoch, and the noted *F1-score* on dev set for this run is also 75.1%. This type of behaviour is a sign of properly trained model which also can generalize well to unseen dataset. And, all the classifications further are done using the *BCE* loss discussed above.

Classification 2: German BERT

German *BERT* (Chan et al. 2019) is another language model which was trained using *German Wikipedia dump*, *OpenLegalData dump*, and *news articles*. The architecture of *German BERT* is based on the original $BERT_{base}$ architecture. The purpose of using *German BERT* model specifically for our classification is because of the reason that our *NTS-ICD10* dataset is in german language, and this model is already pre-trained with a large corpus of german language.

This also makes it reasonable for us to compare the performance of *fine-tuned medical German BERT* language model later so that we can confirm that even *fine-tuning* with small set of datasets can achieve noticeable improvements in the performance.

[7] BCEWithLogitsLoss https://pytorch.org/docs/master/generated/torch.nn.BCEWithLogitsLoss.html

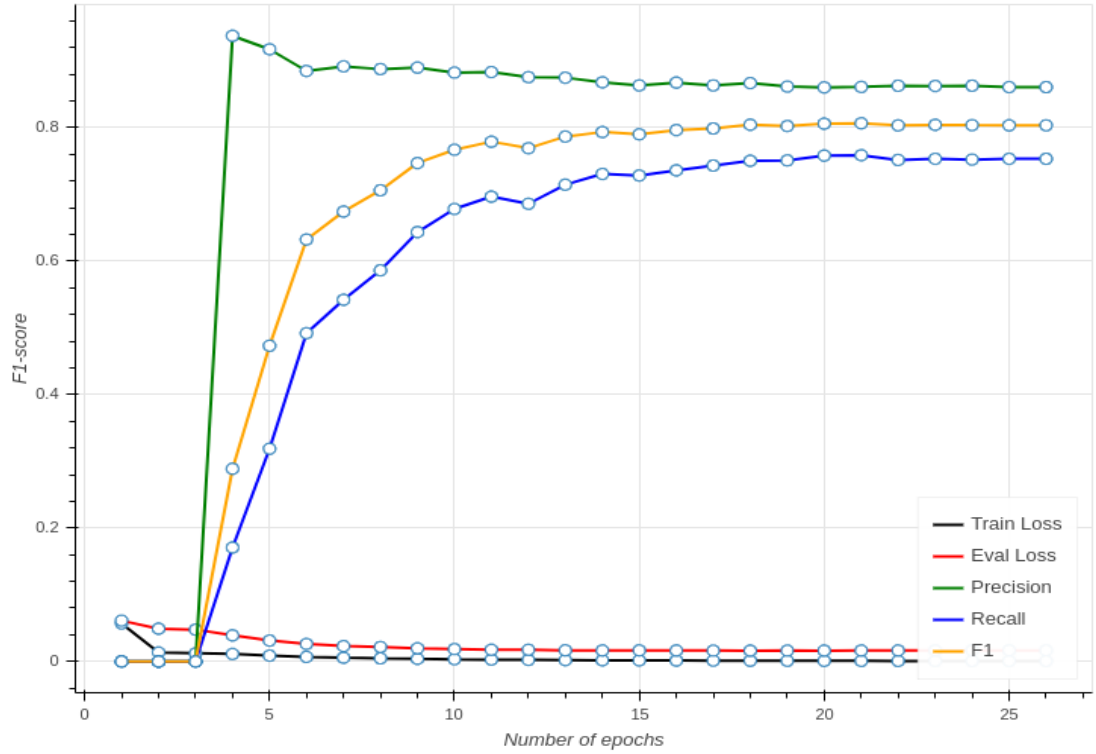


Figure 5.12: German BERT with *BCE* loss for *NTS-ICD10* document classification

Figure 5.12 shows us the classification plot for the *NTS-ICD10* datasets using German *BERT*. And, as we see in the plot, we do not see any signs of overfitting as it was in Figure 5.10. The best score noted for this model was at epoch 19 with *F1-micro* score on dev set of 80.61%. After 19th epoch, the *eval loss* increases with no improvement in the *F1-micro* score. Also, another point to note from the *multi-lingual* and *german BERT* models plot above is that *german BERT* model starts learning quickly than the other and in a stable manner.

This behaviour was a better pointer towards *fine-tuning* the *German BERT* model with additional *medical* datasets so as to make the model more able to learn the *medical* word embeddings and contexts.

Classification 3: Fine-tuned Medical German BERT

In this section, we will make use of our *fine-tuned medical german BERT* to make classification on *NTS-ICD10* dataset. In all the above classification using *BERT*, we used the documents having only the labels which occurred at least 25 times, and only with a maximum sequence length of 256. But, to make this classification more detailed and experimented, we tried three different variants of the datasets as follows:

1. Variant 1: Documents having label occurrence at least 25 and maximum sequence length of 256 (Original implementation)

2. Variant 2: Documents having label occurrence at least 25 and maximum sequence length of 512

Variant 1: Documents having label occurrence at least 25 and maximum sequence length of 256

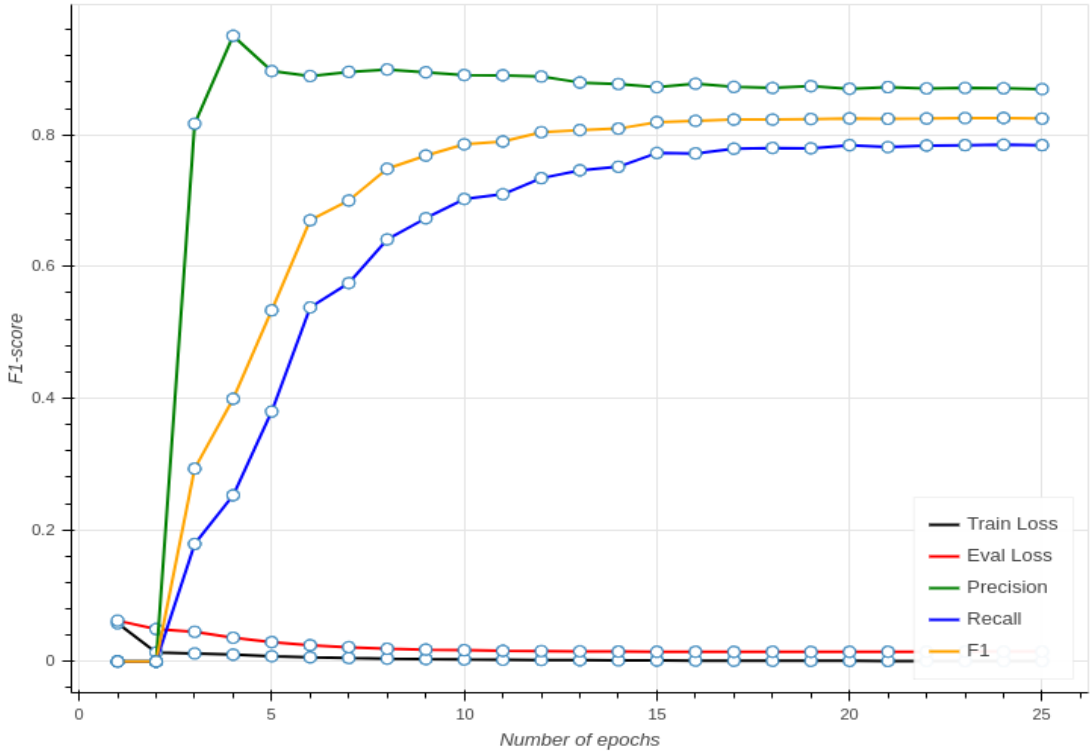


Figure 5.13: Fine-tuned Medical German BERT with maximum sequence length 256 and minimum label occurrence 25 for *NTS-ICD10* document classification

Figure 5.13 shows us the plot of *NTS-ICD10* classification for documents having labels occurrence of at least 25 and a maximum sequence length of 256 although the maximum sequence length supported by *BERT* is 512. If we compare this plot with *German BERT* Figure 5.12, we can see that it starts learning one epoch quicker than it.

This model's plot is quiet similar to Figure 5.12, but the thing to note here is that although the minimum loss was captured in 19^{th} epoch, the *F1-micro* on dev set is 82.42% which is higher than the one captured in Section 5.11.

This model thus generalizes well to some extent and predict the *ICD-10* code missed by *German BERT* model, but still lacks to predict the code for document *6183* as seen in Table 5.13. The reason for the model to not capture the context well could be because it was only trained with a maxium sequence length of 256. But, this was a good sign that *fine-tuning* helps to generalize with the *domain-specific* corpus. So, we train further with a maximum sequence length of 512 which is the maximum sequence length supported by *BERT*.

Variant 2: Documents having label occurence at least 25 and maximum sequence length of 512

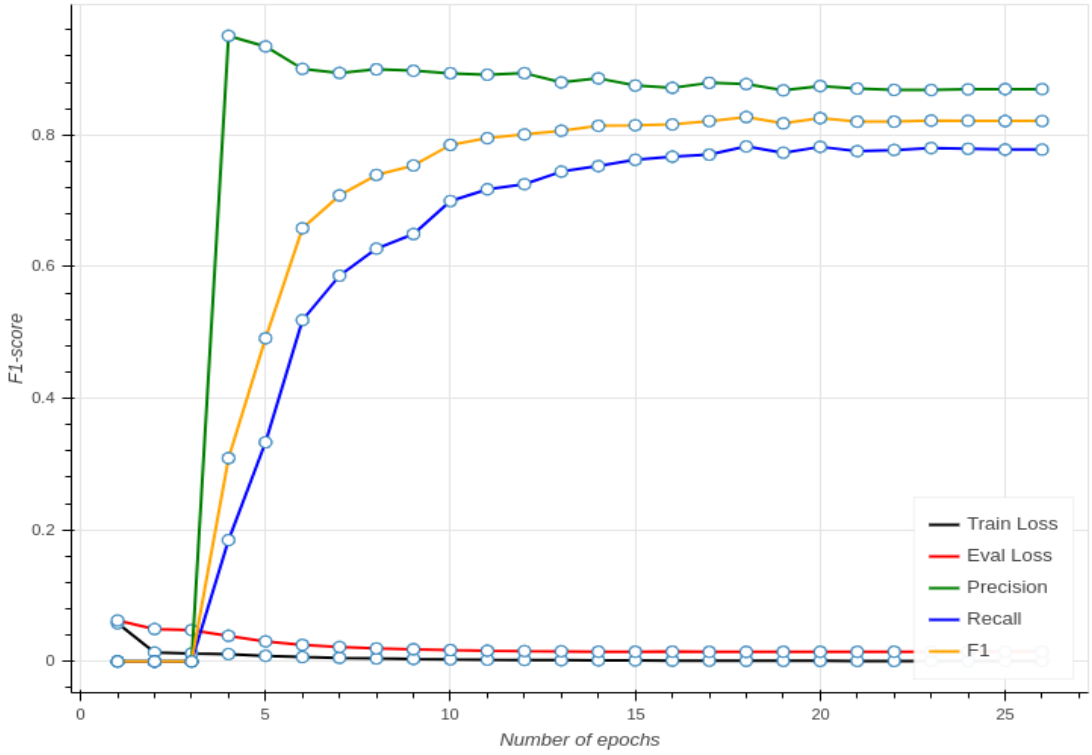


Figure 5.14: Fine-tuned Medical German BERT with maximum sequence length 512 and minimum label occurrence 25 for *NTS-ICD10* document classification

The Figure 5.14 is the plot of training with the same dataset as in the sub-Section 5.12, but it was trained with a maximum sequence length of 512 so as to see the effect on the performance of the model if it was trained with longer sequences. As *BERT* can construct contextual representations from the sequences, we thought of providing it with a larger sequence to capture the context for better learning.

However, compared to the Figure 5.13, the learning in the Figure 5.14 starts one epoch later. The minimum *eval loss* was captured in 17^{th} epoch with the *F1-micro* score on dev set of 82.73%. If we compare results in Section 5.12 and Section 5.13, we see that providing more context improves the performance of the classification model inspite of only selecting the documents having minimum label occurrence of 25.

This model could capture all the *ICD-10* codes related to document *6184* although they are considered as *FP*. But, these codes are meaningful and related to the contents of the documents available in Appendix 6.2, whereas the code *E10-E14* for document 6158 is also increased by 2% and rules out the code *E65-E68* which is not related to the document context as seen in Table 5.14.

5.5 Results

In this section, we will review the results made by the fine-tuned German medical *BERT* model and compare it with other *BERT* models for the experiment tasks that we discussed above in Section 5.3 and Section 5.4.

5.5.1 Masking experiment results

In this section we will go through the sub-word and whole-word masking results made by the baseline *BERT* models and compare it with the fine-tuned German medical *BERT* model.

SubWord masking experiment results

Here we will see the results of sub-word token masking for both nts-sentences and nts-articles by all *BERT* models and compare the masked token predictions made by all those models. Table 5.1 shows the sub-word token masking results for the example mentioned in Section 5.3.1.

Table 5.1: Top-5 sub-word masked tokens prediction result for nts-sentences using German *BERT* model

##ose	**Prob**	**##oi**	**Prob**	**Zentralen**	**Prob**
##otische	34.19	**##oi**	99.60	**zentralen**	99.81
##ose	32.91	##oko	0.12	inneren	0.04
##os	7.28	##ody	0.07	roten	0.03
##ot	6.14	##ore	0.05	menschlichen	0.03
##osit	1.46	##oka	0.04	beteiligten	0.01

Table 5.1 shows us the *top-5* predicted results of the *sub-word masked tokens* for *nts-sentences* with the first row being the original token followed by the predictions, where we can see that out of the 3 tokens *##oi and zentralen* are already better recognized and predicted as the missing token. But, the token *##ose* came in second place and also with less probability score compared to other tokens. This example shows a proper limitation of *German BERT* not being able to effectively recognize the context of the word *multiple sklerose*.

In Appendix 3, we can find a sample prediction for *sub-word* masking on *nts-articles*. The predicted tokens were collected the same way as in for *nts-sentences*. So, for the example mentioned in Appendix 3, let us see the *top-5 result* by the *German BERT* model:

Table 5.2: Top-5 sub-word masked tokens prediction result for nts-articles using German *BERT* model

##in	**Prob**	**##fektion**	**Prob**	**##ische**	**Prob**
[unused_punctuation0]	16.11	##2	9.55	**##ische**	97.91
##typ	5.28	##3	8.67	##ischen	1.46
[unused_punctuation6]	4.91	##1	7.24	##isch	0.28
[unused_punctuation2]	3.46	1	3.65	##ifizierte	0.07
[unused_punctuation1]	2.73	3	3.60	##ologische	0.05

Table 5.2 shows us the *top-5* predicted results of the *sub-word masked tokens* for *nts-articles* with the first row being the original tokens followed by the predictions, where we can see that out of the 3 tokens, only *##ische* is better recognized and predicted as the missing token while the rest tokens are completely missed out.

We can revisit the same thing we did to generate Table 5.1 and 5.2 but with *medical domain fine-tuned BERT* instead of *German BERT*.

Table 5.3: Top-5 sub-word masked tokens prediction result for nts-sentences (Fine-tuned German BERT with medical domain (Section 5.2.3)

##ose	**Prob**	**##oi**	**Prob**	**Zentralen**	**Prob**
##ose	87.27	**##oi**	99.91	**zentralen**	99.83
##itis	7.88	##ody	0.04	menschlichen	0.05
##osis	1.96	##oko	0.01	roten	0.02
##a	1.25	##oka	0.000079	äußeren	0.02
##osa	0.46	##ogra	0.000074	inneren	0.02

Looking at Table 5.3, it shows us the improvement of *medical domain fine-tuned BERT* over *German BERT*. If we specifically see the token *##ose*, there has been a huge increase in the probability score between the two models. And, there is also a small improvement in the score for other two tokens. As the score or the attention was higher for the *piece* word *##ose*, we can say that the model was successfully *fine-tuned*.

Table 5.4: Top-5 sub-word masked tokens prediction result for nts-articles (Fine-tuned German BERT with medical domain (Section 5.2.3)

##in	**Prob**	**##fektion**	**Prob**	**##ische**	**Prob**
##in	64.93	**##fektion**	60.22	**##ische**	91.83
##ein	1.58	##fekt	5.81	##ischen	7.47
-	1.45	##ikation	3.55	##isches	0.19
##ab	1.15	##räglichkeit	1.96	##ischer	0.17
##ins	0.95	##kungen	1.64	##isch	0.16

And, similarly for *nts-articles* in Table 5.4, we also can see the extreme improvement of token predictions as compared to Table 5.2. Similar to the *nts-sentences* in Table 5.2, the

representation were properly learned to build an effective attention model. The results with the fine-tuned model clearly shows that fine-tuning the model with domain specific data improves the performance of the language model.

WholeWord masking experiment results

In this section, we will compare the results of whole-word token masking for both nts-sentences and nts-articles by all *BERT* models and compare the masked token predictions made by all those models. The example of this experiment for nts-sentences is mentioned in Section 5.3.2.

Table 5.5: Top-10 whole-word masked tokens prediction result for nts-sentences using German *BERT* model

Sk	**Prob**	**##ler**	**Prob**	**##ose**	**Prob**
Hyper	4.07	##ipl	4.41	##itis	27.82
Neur	3.41	##ap	1.45	**##ose**	11.80
Mult	2.98	##pat	1.18	##ie	3.01
Bron	2.17	**##ler**	1.17	##rose	2.93
Leber	2.07	##arth	1.13	##heimer	2.68
Muskel	1.94	##kul	1.05	##enz	2.66
An	1.92	##rel	1	##störung	2.23
Diabetes	1.88	##enz	0.9	##mie	1.75
Ast	1.82	##ond	0.89	##a	1.57
Pro	1.81	##io	0.87	##ation	1.51

In Table 5.5, we can see a sample prediction for the tokens while using *whole word* masking. It will be relatively difficult for the model to predict the *full-word* rather than only the *piece of word* because the attention is higher to the *word piece* when followed by its *initial piece* which results in a successful prediction of the *sub-word* tokens. As from the result in Table 5.5, we can see that the model cannot recognize the starting piece following the token *multiple*, and results in the *Sk* token to be present in 11^{th} position with a probability of *1.72*, and the rest are already present within the top-10 position but with less probability scores.

However, *nts-articles* has shown a better score in Figure 5.7, let us see a sample prediction for *whole-word* masking on *nts-articles* which is made available in Appendix 4.

Table 5.6: Top-10 whole-word masked tokens prediction result for nts-articles using German *BERT* model

Virus	**Prob**	**##in**	**Prob**	**##fektion**	**Prob**
[unused_punctuation2]	11	[unused_punctuation0]	27.10	##2	9.59
[unused_punctuation0]	3.86	[unused_punctuation6]	4.46	##1	7.01
[unused_punctuation5]	3.61	Typ	1.75	##3	6.76
und	3.42	##P	1.59	##5	3.41
Virus	2.75	Virus	1.54	2	3.12
[unused_punctuation6]	2.32	[unused_punctuation1]	1.25	1	2.95
[unused_punctuation3]	1.96	Infektion	1.02	3	2.93
[unused_punctuation1]	1.95	##I	0.81	##4	2.89
[unused_punctuation4]	1.39	[unused_punctuation2]	0.76	##6	2.47
Typ	1.14	[UNK]	0.71	4	2.42

The result from Table 5.6 shows that although the scores here seem better than in *nts-sentence*, it also has large amount of sentences and hence more tokens and context to predict from. But, still out of the 3 tokens, only 1 has been successfully found with less probability, so it could be a high likely that after *fine-tuning* this model with *medical documents*, the scores can improve a little.

We can now verify the effects of *fine-tuned* model for *whole-word* masking as we did for *sub-word* masking in Table 5.3 and 5.4. The models performance improved by 46.15% on the *top-10* predicted tokens and by 28.57% on *top-100* predicted tokens for nts-sentences. This is a good sign that *fine-tuning* a model using *domain specific* datasets has improved the overall model performance.

We can now revisit the score tables Table 5.5 and 5.6, and check if *fine-tuning* model can really give us some correct tokens now. Table 5.7 shows the performance of the *fine-tuned* model for *whole-word* masked tokens, and we can see that the *fine-tuned* model really learns the representation for the token *Sk* where the *German BERT* model failed. Also, it shows that the attention for *medical terms* has gradually increased but not by so much.

Table 5.7: Top-10 whole-word masked tokens prediction result for nts-sentences (Fine-tuned German BERT with medical domain (Section 5.2.3)

Sk	**Prob**	**##ler**	**Prob**	**##ose**	**Prob**
Neur	7.14	##erkran	4.14	**##ose**	24.93
My	4.57	**##ler**	4.14	##itis	11.48
Ast	4.53	##arth	4.13	##rose	9.66
An	3.39	##kul	1.88	##ie	8.26
Hyper	2.99	##mat	1.67	##enz	4.86
Sk	2.80	##gener	1.31	##störung	4.01
Chron	2.14	##zir	1.26	##sucht	2.82
Fi	1.87	##entz	1.26	##ase	2.29
Col	1.54	##ond	1.16	##ung	2.13
Diabetes	1.32	##ph	1.14	##ation	1.45

Table 5.8 predictions were not so helpful as that of Table 5.6 but, the only slightest improvement was the former model did not predict any *[unused_punctuation]* token. It gave some *word-piece* tokens but none of them were a meaningul token match. There was a 9.09% of improvement in the token *Virus* with the *fine-tuned* model compared to the other.

Table 5.8: Top-10 whole-word masked tokens prediction result for nts-articles (Fine-tuned German BERT with medical domain (Section 5.2.3)

Virus	**Prob**	**##in**	**Prob**	**##fektion**	**Prob**
Die	5.64	Typ	2.08	##2	1.71
Virus	3	##P	1.94	##1	1.70
-	2.35	##M	1.43	##e	1.60
.	2.2	.	1.40	##3	1.16
Diese	2.14	Virus	1.25	.	1.15
und	2.01	##E	1.17	##ie	0.96
Der	1.93	##N	1.06	Re	0.87
Eine	1.69	##krebs	0.86	##o	0.82
Typ	1.68	Vir	0.77	Virus	0.76
H	1.57	und	0.77	##a	0.76

The *fine-tuned* model did improve the *token attention* and *masked prediction* for *sub-word masking*, but quiet still the attention mechanism has not been so efficent for *whole-word masking*.

5.5.2 NTS-ICD-10 classification results

In this section, we will review the classification results made by baseline *BERT* models and compare it with the fine-tuned *BERT* model. Figure 3.1 shows the distribution of dataset that we used to train, validate, and predict respectively. And, similarly Table 5.9

lists out all the scores that we achieved for those distributions using the baseline models. All the metrics for *P*, *R*, and *F1*-score are calculated as micro averaged.

Baseline models classification results

Table 5.9 shows the scores achieved for the classification of *NTS-ICD-10* documents using several baseline model. As we also have some general *ML* algorithms in the score table, we are only using it as a baseline score while our focus is towards the *BERT* model: *multi-lingual* and *german BERT* in this case. Comparing the *multi-lingual* and *german BERT* here, *german BERT* seems to be more stable in the *P* and *R* scores in both *development* and *test* dataset.

Table 5.9: Baseline scores for *NTS-ICD-10* document classification.

	Dev			Test		
	P	R	F1	P	R	F1
Linear SVC	0.812	0.704	0.754	0.847	0.638	0.728
Logistic Regression	0.722	0.761	0.741	0.742	0.716	0.728
Fasttext	73.04	32.95	45.42	66.58	31.01	42.31
Multi-lingual BERT	86.21	66.44	75.1	66.4	84.5	74.4
German BERT	86.04	75.82	80.60	85.3	75.8	80.3

As per the classification by *multi-lingual* model seen in Section 5.4.4, it has all the *ICD-10* codes predicted correctly along with a series of other *ICD-10* codes also in there considered as *FP*. We will not go through the *FP* codes here, but will later compare it with *fine-tuned BERT* codes in Section 5.5.2 and deduce if they were meaningful. *German BERT*, on the other hand does not have the same amount of *FP* but instead introduces *FN* and even fail to predict any *ICD-10* code in the document *6198*.

Table 5.10: NTS-ICD-10 code predictions with BERT multilingual model

Doc	**Actual**	**Fine-tuned-512**
6198	(XVII, Q80-Q89)	(XVII, Q80-Q89, Q90-Q99)
6184	(VI, G35-G37)	(VI, G35-G37, I, A30-A49, XIII)
6158	(E65-E68, IV)	(E65-E68, IV, E10-E14)
6183	(VII, H15-H22)	(VII, H15-H22, H25-H28, H30-H36, H40-H42, H46-H48, H49-H52, H53-H54, H55-H59)

As seen in Table 5.10, we can see the predicted *ICD-10* codes where we can see although the *actual labels* also exist in the *predicted* ones, but there is also the high presence of *FP*. Also referring to the Table 5.9, *multilingual* model shows the presence of high *false positives* and *false negatives* which has a negative impact in *F1-micro* score. *Multilingual* model is trained with the texts from multiple languages as discussed in Section 5.4.4

which could be the reason for the model to generalize well. We will not get into more details with this model as we are more focused with the result in *German BERT* model and the *fine-tuned* model.

Table 5.11: NTS-ICD-10 code predictions with German BERT model

Doc	Actual	German
6198	(XVII, Q80-Q89)	no-predictions-here
6184	(VI, G35-G37)	(VI **0.95**, G35-G37 **0.92**)
6158	(E65-E68, IV)	(E10-E14 **0.89**, E65-E68 **0.91**, IV **0.99**)
6183	(VII, H15-H22)	(XIX **0.51**)

In Table 5.11, we can find the predictions made by *German BERT* for the same documents as done in 5.10 along with the probability of the occurence of *ICD-10* codes. As we know that *German BERT* is trained with *wikipedia and books corpus* which mostly does not contain *medical terms*, and hence as a result we can see that this model could not generalize well with the *NTS* data. As a result this model serves us with both *FP* and *FN*, and also some cases where it is not able to conclude any *ICD-10* code. The document *6158* has a *FP ICD-10* code *E10-E14* which is somewhat meaningul while looking at the text in document *6158*. The text segments responsible for this classification has been highlighted *bold* in the Appendix 6.3. The document *6158* is about **Influence of obesity on mitochondrial dysfunction** and related to the diagnosis of **diabetes mellitus** which belongs to the *ICD-10* code *E11.-* which further falls under the group *E10-E14*[8]. As the group *E65-E68* is only about *obesity and other overeating*, so, *E10-E14* is also considered as a reasonable *ICD-10* group for this document.

The *ICD-10* code *VII* is considered as *FN* here with a probability of *0.46* which was almost near to the threshold of 0.5. This model has almost caught the *ICD-10* codes related to the documents, but to be more general, let us find out if the *fine-tuned* model really improves this part.

Fine-tuned model classification results

Table 5.12 shows the scores achieved for the classification of *NTS-ICD-10* documents using *medical fine-tuned BERT* models . Comparing the *F1* score of *medical fine-tuned BERT* with *german BERT*, it has a increase of 2.25%. This model has improved a little bit in predicting the codes that was not discovered by *german BERT* for document *6198* but also has increased number of *FP* and *FN*. While we saw a slight improvement in the predictions, this model is not still able to generalize well with the texts. And, we thought this was due to the value of *maximum sequence length* the model was being trained with. This model was trained with a *maximum sequence length* of *256* which can be seen in

[8]Diabetes-with-obesity https://www.dimdi.de/static/de/klassifikationen/icd/icd-10-gm/kode-suche/htmlgm2019/block-e10-e14.htm

Section 5.12, while the *maximum sequence length* supported by *BERT* is 512. So, we trained the model again with a *maximum sequence length* of 512 which can be seen in Section 5.13, and it has a *F1* score gain of 2.64% with *german BERT* and 0.38% with the *fine-tuned* model with a *maximum sequence length* of 256 (Section 5.12).

Table 5.12: Medical Fine-tuned BERT scores for *NTS-ICD-10* document classification.

	Dev			Test		
	P	R	F1	P	R	F1
Fine-tuned medical BERT (Section 5.12)	87.41	77.97	82.42	87.51	77.90	82.43
Fine-tuned medical BERT (Section 5.13)	87.75	78.26	82.73	86.65	77.13	81.62

This model has a high number of *TP* with a less number of *FP* and *FN* as compared to the previous models in Section 5.4.4 and 5.11. It also has a increased probability score for the *ICD-10* code predicted, while the *fp* codes in documents *6184* and *6158* are also relevant which are discussed in Section 5.12 and 5.13. As we go inside the documents and evaluate the terms, this model seems to generalize well with all the *medical terms* and learns the hidden represenations strongly given more context.

As seen in Section 5.12, we performed training using fine-tuned models in two different ways. One was with utilizing the maximum sequence length of 256 and the other was 512. Here, we will see the results with two different combinations and see if the mazimum sequence length affects the performance of the model for prediction *ICD-10* codes.

Table 5.13: NTS-ICD-10 predictions with medical domain fine-tuned BERT model (max-sequence-length: 256)

Doc	**Actual**	**Fine-tuned-256**
6198	(XVII, Q80-Q89)	(XVII **0.56**)
6184	(VI, G35-G37)	(A30-A49 **0.87**, I **0.94**)
6158	(E65-E68, IV)	(E10-E14 **0.94**, E65-E68 **0.90**, IV **0.99**)
6183	(VII, H15-H22)	(XIX **0.73**)

The Table 5.13 shows that *fine-tuning* has improved the classification purpose for *NTS* documents. Concerning document *6198*, this model could predict the code *XVII* with a score of 0.56 which is a good sign that the model could atleast point out the *chapter* code, where it also has increased probability score for the group *E10-E14* compared to that in 5.11. But, if we see for document *6184*, we can see great number of *FP* here, where Section 5.11 could exactly give all *TP*. The document *6184* as described in Appendix 6.2

mentions about both *Multiple Sklerose* and *Sepsis* model. *German BERT* lacks to generalize to *sepsis* from *group A30-A49*[9] while this model lacked to generalize to *multiple sklerose* from *group G35-G37*[10].

Table 5.14: NTS-ICD-10 predictions with medical domain fine-tuned BERT model (max-sequence-length: 512)

Doc	**Actual**	**Fine-tuned-512**
6198	(XVII, Q80-Q89)	(XVII **0.77**)
6184	(VI, G35-G37)	(VI **0.93**, I **0.91**, A30-A49 **0.75**)
6158	(E65-E68, IV)	(IV **0.99**, E10-E14 **0.98**)
6183	(VII, H15-H22)	(VII **0.95**)

Table 5.14 shows us the better classification result as compared to the Table 5.13. First for document *6198*, the score for the code *XVII* has increased by 23%, and another good thing is that it could retrieve the *chapter code VII* for document 6183 with a score of 95%. It also shows that giving the whole sequence as a input, the model can really capture the hidden representations well which is proved by the result of the documents *6184* and *6158*.

[9]Sepsis https://www.dimdi.de/static/de/klassifikationen/icd/icd-10-gm/kode-suche/htmlgm2019/block-a30-a49.htm

[10]Multiple-Sklerose https://www.dimdi.de/static/de/klassifikationen/icd/icd-10-gm/kode-suche/htmlgm2019/block-g35-g37.htm

Chapter 6

Conclusion and Future Work

6.1 Review

In this thesis work, we presented an example of transfer learning of the recent *BERT* model to the medical domain. Based on the related literature, we proceeded with an approach that doesn't require the *pre-training* of the whole model from scratch but instead initializes the weight of the model with an existing one. For this purpose, we chose *German BERT* (Chan et al. 2019) in our case. This approach is convenient for several reasons:

1. Suitable, when the volume of available textual corpora is not large enough to meet the specifications for *pre-training* an entirely new model.
2. Suitable regarding the computational time which is less compared to the time required to *pre-train* whole new model.

For transfer learning to work, it is better that the source domain and the target domain share a common base, which in our case is the understanding of the *German* language. So, *German BERT* was the suitable model instead of the *Multilingual BERT* model regarding the pre-training performance. The creation of a custom vocabulary should help *German Medical BERT* better than the language by weighting the additional embeddings more accurately. *SciBERT*, the pre-trained *BERT* model for scientific texts (Beltagy, Lo, and cohan 2019), improves around 60% on F1-sore across all the datasets on average. This shows additional improvement while creating your vocabulary for *SciBERT*.

For our case, the original vocabulary worked out better than the *German BERT* model when we *fine-tuned* it with additional *medical articles* which can be found in Section 3.4. The performance of the classical baseline models like *SVC, Logistic Regression* are better, but lower than the *BERT* models. It is also surprising that *Fasttext* could not reach up to the score of *svc or regression*. *LinearSVC* somehow outperforms *multilingual BERT* by a bit, and *German BERT* also generalizes the language and overcomes the performance by 7.27% on development set and 7.93% on test set. Hence, this was also the reason of choosing *German BERT* model instead of *multilingual BERT* model for fine-tuning.

Fine-tuned medical BERT model has shown to be better for both tasks: *token mask prediction* and *classification*. However, it could not generalize well for *token mask* prediction in articles, but it was better enough to understand the context in sentences and predict the masks. We masked the tokens in two ways: *sub-word* and *whole-word*. Here, *sub-word* masking performed better than *whole-word* considering that it is easier to predict the second piece of the token given the first piece. But, it was difficult enough for the model to get all the token pieces from the *whole word* masks, however, it performed better than the *German BERT* model. If we see at the *classification task*, *fine-tuned German BERT* model performed 2.64% better than the *German BERT* model. This is an improvement and indication that *fine-tuning* the model with domain-specific data helps to get better results.

After performing all the masking experiments in Section 5.3 and *NTS-ICD-10* document classification in Section 5.4, we can now go ahead and answer the research questions mentioned in Section 1.3. The masking experiment however performed with the fine-tuned model is also known as the target task and the classification task further performed to train a new model is known as a downstream task. The research question can now be answered as follows respectively:

1. As we now know that our target domain is medical datasets in the German language, we performed fine-tuning on the German *BERT* using the German medical articles that we collected from the web and the statistics on the fine-tuning dataset can be found in Section 3.4. Here, our target task is to predict the masked tokens as the underlying model is trained using the *MLM* objective. After performing the masked tokens experiment in Section 5.3 and comparing the result of the German *BERT* model with the fine-tuned medical German *BERT* model in Section 5.5.1, we can conclude that fine-tuning on domain-specific data improves the quality of the language model on the target domain as we have seen improved attention to the medical terms by the fine-tuned model and the masked tokens have been predicted with better scores than the German *BERT* model.

2. After being convinced that fine-tuning has improved the performance for the target domain, it is now time to see if it goes the same way for the classification task. In order to achieve this, we performed the *NTS-ICD-10* document classification using various baseline *BERT* models and the fine-tuned *BERT* model in Section 5.4 and later we compare the *ICD-10* codes prediction in Section 5.5.2. After comparing the results in Table 5.11, Table 5.13, and Table 5.14, we can see that although German *BERT* has predicted some *ICD-10* codes, it lacked some generalization and also missed out some codes for document 6198. But, this problem has been handled by the fine-tuned model where the code for document 6198 has been predicted and the prediction score has even increased in the model trained with a sequence length of 512. Another example of extracting the code is for document 6183 where

the German *BERT* and the model with a sequence length of 256 fails to do so. Besides it, with the remaining two documents 6184 and 6158, we can see a better generalization in predicting the *ICD-10* codes by the model with a sequence length of 512, as it has selectively captured the context and provided with more accurate *ICD-10* code representations.

3. By this point, we have successfully answered two of our research questions. The third question is also somewhat similar to the second question. The second question talks about if fine-tuning improves the performance on downstream tasks while the third question is curious if improvement in language model also implies improvement on downstream tasks. And, here from the evaluation on the first question's answer, we know that fine-tuning has improved the performance of the language model as the prediction of the masked tokens has improved after fine-tuning it with the target domain. Similarly, from the evaluation on the second question answer, we can also infer that this fine-tuned or improved language model has improved the performance on the downstream task which in here was document classification. But, we cannot say that the performance would be the same for all types of downstream tasks without having it experimented. It could imply the same for all types of downstream tasks like *NER*, *QA*, and others, but we only evaluated it for document classification as this was within the scope of our thesis.

6.2 Discussion

As the *German BERT* model has been trained using the latest *German Wikipedia dump* and *openLegalData dump* (Chan et al. 2019), we thought that it would not be able to generalize well with *medical* data. But, the result of *German BERT* model was already better than the classical baseline algorithms. This does not mean it was already handling the medical terms with its highest attention, but only with slightest improvement over the masking performance when compared with *wikipedia texts*. So, we believed that *fine-tuning* this model with some domain-specific data (medical data) would give more attention to the medical terms and hence have some improvement over the *token masking* performance. Here, our main goal is to classify the *NTS-ICD-10* documents containing the medical experiments.

But, *MLM* is also used in *fine-tuning* objective as done during *pre-training*. So, we also found out that *fine-tuning* did increase attention to medical terms, and hence, the masking score was higher as the masked word was already being predicted in a lower position. And, hence this increasing attention in medical terms led to a better score in the classification with a significant increase in the *precision* and *recall*. And, also we found that when the model was trained with a sequence length of *512*, the model was capturing the context better than when trained with a sequence length of *256*. Therefore, this also leads

to the weakness of *BERT* which is the maximum sequence length. As seen for many tasks *512* tokens is sufficient, but as mentioned before where longer sequence length gave better results, we only think that if the sequence length was not limited maybe we would have more convincing results.

The *fine-tuning* process saves a considerable amount of time and yields acceptable results with smaller model size and less computation time. But, as a consideration of research, if the model keeps growing this would not be a democratic situation, where only expensive hardware will be able to perform experiments. However, *fine-tuning* would produce much better results when provided with even more data than the ones already there which would not be so resource consuming. In contrast, *AI* applied to the medical domain is growing exponentially which are experimenting with various *NLP* and *LM* procedures to get the breakthrough. The main limitation behind working freely with medical data is the less availability of open medical data in *German* language.

6.3 Future Work

As discussed above, data scarcity is a major obstacle in performing *LM* on domain-specific data. If there would be more available data and not so expensive resources, we could pre-train from scratch to produce the *German Medical BERT* model to compare with the current one. But, for now, the other possible improvement using language *fine-tuning* would be training further with more medical data and evaluate again. Extending the contextualized word representations with specific world knowledge about medical language would be interesting follow-up research.

Another main thing to try in the future would be to filter out the labels in the *NTS-ICD-10* documents. Each document is assigned with a collection of *ICD-10* codes which consists of multiple levels of codes in it like *II, C00-C97, C00-C75, C15-C26* where the main or the low-level group is *C15-C26*, and the rest are only the parent of each other respectively. So, in this case, referring to Figure 3.5, we can say that when a child group is present it is also often followed by its parent group. The notion of the child group and parent group can be revisited in Figure 3.2. So, instead of training with all those class combinations, if we train with only the low-level *ICD-10* group code present in the *NTS-ICD-10* document, the model would have a fewer number of class labels to deal with which would result in the model to generalize well.

This would be an ideal experiment to be conducted in the future where we can handle some hierarchical structure of *ICD-10*.

Reference List

Adhikari, Ashutosh et al. (2019). "DocBERT: BERT for Document Classification". In: *cs.CL* 3, pp. 1–4. arXiv: 1904.08398.

Alsentzer, Emily et al. (2019). "Publicly Available Clinical BERT Embeddings". In: *cs.CL* 3, pp. 1–7. arXiv: 1904.03323.

Amin, Saadullah et al. (2019). "MLT-DFKI at CLEF eHealth 2019: Multi-label Classification of ICD-10 Codes with BERT". In: *CLEF eHealth 2019*, pp. 1–8. URL: https://www.dfki.de/fileadmin/user_upload/import/10515_paper_67.pdf.

Asch, Vincent Van (2013). "Macro- and micro-averaged evaluation measures". In: *Semantic Scholar*. URL: https://pdfs.semanticscholar.org/1d10/6a2730801b6210a67f7622e4d192bb309303.pdf.

Bahdanau, Dzmitry, KyungHyun Cho, and Yoshua Bengio (2016). "NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE". In: *cs.CL* 7, pp. 1–8. arXiv: 1409.0473.

Beltagy, Iz, Kyle Lo, and Arman cohan (2019). "SciBERT: A Pretrained Language Model for Scientific Text". In: *cs.CL* 3, pp. 1–6. arXiv: 1903.10676.

Bert, Bettina et al. (2017). "Rethinking 3R strategies: Digging deeper into Animal-TestInfo promotes transparency in vivo biomedical research". In: *PLoS biology 15(12): e2003217* 15, pp. 1–8. DOI: 10.1371/journal.pbio.2003217. URL: https://doi.org/10.1371/journal.pbio.2003217.

Brants, Thorsten (2003). "Natural Language Processing in Information Retrieval". In: *http://citeseerx.ist.psu.edu/*, pp. 1–10. URL: http://www.cnts.ua.ac.be/clin2003/proc/03Brants.pdf.

Chan, Branden et al. (2019). "German BERT". In: URL: https://deepset.ai/german-bert.

Chen, Dawn, Joshua C. Peterson, and Thomas L. Griffiths (2017). "Evaluating vector-space models of analogy". In: *cs.CL* 1, pp. 1–3. arXiv: 1705.04416.

Cun, Yann le (1988). "A Theoretical Framework for Back-Propagation". In: pp. 1–3. URL: http://yann.lecun.com/exdb/publis/pdf/lecun-88.pdf.

Dalal, Mita K. and Mukesh A. Zaveri (2011). "Automatic Text Classification: A Technical Review". In: *International Journal of Computer Applications* 28, pp. 1–2. URL: https://www.researchgate.net/profile/Mukesh_Zaveri/publication/266296879_Automatic_Text_Classification_A_Technical_Review/links/54e74a0a0cf2b199060ae1c5/%20Automatic-Text-Classification-A-Technical-Review.pdf.

Devlin, Jacob et al. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *cs.CL* 2, pp. 1–5. arXiv: 1810.04805.

Do, Chuong B. and Andrew Y. Ng (2006). “Transfer learning for text classification”. In: *papers.nips.cc*, pp. 1–6. URL: https://papers.nips.cc/paper/2843-transfer-learning-for-text-classification.pdf.

Fortuny, Enric Junque de, David Martens, and Foster provost (2013). “PREDICTIVE MODELING WITH BIG DATA”. In: *https://www.liebertpub.com/*, pp. 1–9. URL: https://www.liebertpub.com/doi/pdf/10.1089/big.2013.0037.

Geneva, World Health Organization (2019). “International Statistical Classification of Diseases and Related Health Problems (Tenth Revision)”. In: 2, pp. 1–13. URL: https://ec.europa.eu/cefdigital/wiki/display/EHSEMANTIC/WHO+ICD-10+The+International+Statistical+Classification+of+Diseases+and+Related+Health+Problems+10th+Revision.

Ghamrawi, Nadia and Andrew McCallum (2005). “Collective Multi-Label Classification”. In: *Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 1–2. URL: https://doi.org/10.1145/1099554.1099591.

Howard, Jeremy and Sebastian Ruder (2018). “Universal Language Model Fine-tuning for Text Classification”. In: *cs.CL* 5, pp. 1–9. arXiv: 1801.06146.

Jing, Kun and Jungang Xu (2019). “A Survey on Neural Network Language Models”. In: *cs.CL* 2, pp. 1–4. arXiv: 1906.03591.

Joulin, Armand et al. (2016). “Bag of Tricks for Efficient Text Classification”. In: *cs.CL*, pp. 1–4. arXiv: 1607.01759.

Jurafsky, Daniel and James H. Martin (2019). “Sequence Processing with Recurrent Networks”. In: pp. 2–12. URL: https://web.stanford.edu/~jurafsky/slp3/9.pdf.

K, Soymya George and Shibily Joseph (2014). “Text Classification by Augmenting Bag of Words (BOW) Representation with Co-occurrence Feature”. In: *IOSR Journal of Computer Engineering (IOSR-JCE)* 16, pp. 1–2. URL: https://www.researchgate.net/profile/Shibily_Joseph/publication/271294957_Text_Classification_by_Augmenting_Bag_of_Words_BOW_Representation_with_Co-occurrence_Feature/links/5a38f1560f7e9b7c48701084/Text-Classification-by-Augmenting-Bag-of-Words-BOW-Representation-with-Co-occurrence-Feature.pdf.

Lazreg, Mehdi Ben, Morten Goodwin, and Ole-Christoffer Granmo (2016). “Deep Learning for Social Media Analysis in Crises Situations”. In: 4. URL: https://ep.liu.se/ecp/129/004/ecp16129004.pdf.

Lee, Jinhyuk et al. (2019). “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *cs.CL* 4, pp. 1–7. arXiv: 1901.08746.

Mandelbaum, Amit and Adi Shalev (2016). “Word Embeddings and Their Use In Sentence Classification Tasks”. In: *cs.LG*, pp. 1–4. arXiv: 1610.08229.

Mikolov, Tomas et al. (2013). “Distributed Represenations of Words and Phrases and their Compositionality”. In: *cs.CL* 1, pp. 1–7. arXiv: 1310.4546.

missinglink.ai (n.d.). “The Complete Guide to Artificial Neural Networks: Concepts and Models”. In: *https://missinglink.ai/guides/neural-network-concepts/complete-guide-artificial-neural-networks/ [Accessed 18.07.2020]* ().

Murphy, Jo (2012). “EFFECTIVE NON-TECHNICAL SUMMARIES FOR ENVIRONMENTAL IMPACT ASSESSMENT”. In: *iema.net*, pp. 1–2. URL: https://www.iema.net/download-document/17875.

Nasteski, Vladimir (Dec. 2017). “An overview of the supervised machine learning methods”. In: *HORIZONS.B* 4, pp. 4–8. DOI: 10.20544/HORIZONS.B.04.1.17.P05.

Nwankpa, Chigozie Enyinna et al. (2018). “Activation Functions: Comparison of Trends in Practice and Research for Deep Learning”. In: *cs.LG* 1. arXiv: 1811.03378.

Peters, Matthew E. et al. (2018). “Deep contextualized word representations”. In: *cs.CL* 2, pp. 1–8. arXiv: 1802.05365.

Pilz, Anja (2015). “Entity Linking to Wikipedia (Grounding entity mentions in natural language text using thematic context distance and collective search)”. In: pp. 56–59. URL: http://hss.ulb.uni-bonn.de/2016/4240/4240.pdf.

Pires, Telmo, Eva Schlinger, and Dan Garrette (2019). “How multilingual is Multilingual BERT?” In: *cs.CL* 1, pp. 1–6. arXiv: 1906.01502.

Popowich, Fred (2005). “Using Text Mining and Natural Language Processing for Health Care Claims Processing”. In: *SIGKDD Explorations* 7, pp. 1–2. URL: http://www.kdd.org/exploration_files/9-Popowich.pdf?searchterm=Dollar+Volume+Definition+of+Dollar%20+Volume.

Ramos, Juan (2003). “Using TF-IDF to Determine word Relevance in Document Queries”. In: *Semantic Scholar*, pp. 1–2. URL: https://pdfs.semanticscholar.org/b3bf/6373ff41a115197cb5b30e57830c16130c2c.pdf.

Read, Jesse et al. (2009). “Classifier Chains for Multi-label Classification”. In: *ECML PKDD* 5782, pp. 1–5. URL: https://link.springer.com/content/pdf/10.1007%2F978-3-642-04174-7_17.pdf.

Roesch, Jared et al. (2018). “Relay: A New IR for Machine Learning Frameworks”. In: *cs.PL* 1, pp. 1–8. arXiv: 1810.00952.

Rosenstein, Michael t., Zvika Marx, and Leslie Pack Kaelbling (2005). “To Transfer or Not To Transfter”. In: *http://web.engr.oregonstate.edu/*, pp. 1–4. URL: http://web.engr.oregonstate.edu/~tgd/publications/rosenstein-marx-kaelbling-dietterich-hnb-nips2005-transfer-workshop.pdf.

Russell, W.M.S. and R.L. Burch (1959). “The Principles of Humane Experimental Technique”. In: URL: https://caat.jhsph.edu/principles/chap4d.

Sänger, Mario et al. (2019). "Classifying German Animal Experiment Summaries with Multi-lingual BERT at CLEF eHealth 2019 Task 1". In: *Researchgate*, pp. 2–8. URL: https://www.researchgate.net/profile/Ulf_Leser/publication/334989501_Classifying_German_Animal_Experiment_Summaries_with_Multi-lingual_BERT_at_CLEF_eHealth_2019_Task_1/links/5d49679392851cd046a5a5fb/Classifying-German-Animal-Experiment-Summaries-with-Multi-lingual-BERT-at-CLEF-eHealth-2019-Task-1.pdf.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *cs.CL* 3, pp. 1–6. arXiv: 1409.3215.

Tan, Chuanqi et al. (2018). "A Survey on Deep Transfer Learning". In: *cs.LG* 1, pp. 1–2. arXiv: 1808.01974.

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *cs.CL* 5, pp. 1–9. arXiv: 1706.03762.

WHO, DIMDI (2019). "International statistical classification of diseases and related health problems (10th revision) German modification". In: URL: https://www.dimdi.de/static/de/klassifikationen/icd/icd-10-gm/kode-suche/htmlgm2019/index.htm.

Yamashita, Rikiya et al. (2018). "Convolutional neural networks: and overview and application in radiology". In: *springer*, pp. 1–3. URL: https://link.springer.com/content/pdf/10.1007/s13244-018-0639-9.pdf.

Yang, Yimin (1998). "An Evaluation of Statistical Approaches to Text Categorization". In: *INRT Journal*, pp. 6–9. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.2485&rep=rep1&type=pdf.

Zhang, Yin, Rong Jin, and Zhi-Hua Zhou (2010). "Understanding bag-of-Words Model: A Statistical Framework". In: *International journal of Machine Learning and Cybernetics*, pp. 1–2. URL: https://www.researchgate.net/profile/Rong_Jin/publication/226525014_Understanding_bag-of-words_model_A_statistical_framework/links/554b968f0cf29752ee7cc15b/Understanding-bag-of-words-model-A-statistical-framework.pdf.

Zhou, Peng et al. (2016). "Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling". In: *cs.CL* 1, pp. 1–8. arXiv: 1611.06639.

Appendices

Tools and Environment

1 Software

Being already experienced with *Python* programming languaage, we found it easier that most of the deep learning research and production has been conducted with it. And, also considering the machine learning framework, we chose *PyTorch* because of its simpler approach and faster learning curve. At the moment of writing this paper, *PyTorch* was already compatible for fine-tuning and had various implementations for creating a new language model. It is also worth mentioning that using *PyTorch* as a foundation, we also used *Hugging Face* to perform our fine-tuning experiments. [1] [2] [3]

2 Hardware

Deep learning computations are usually heavy and require more memory as well as time. So, to perform some evaluation experiments we used *Google Colab's GPU*, *Kaggle GPU notebook*, and for further evaluations like fine-tuning, we used *AWS* instances by starting some batch jobs in it. The remote *AWS* machine used Ubuntu 16.04 while, the local machine used Ubuntu 18.10. [4]

[1] Python https://www.python.org/
[2] Pytorch https://pytorch.org/
[3] HuggingFace https://huggingface.co/
[4] Google Colab https://colab.research.google.com

Learning Parameters

Hyperparameters are the set of parameters which is set before starting the learning process of a machine learning model. Some parameters like *learning rate*, *number of epochs*, *train batch size*, *eval batch size*, and others can be adjusted during the beginning of training.

1 Fine-tuning parameters

BERT has specific suggestions of learning parameters for *fine-tuning* like, learning rate of either (*2e-5, 3e-5 or 4e-5*), epochs of (2, 3, or 4), and a batch size of (16, 32, or 64) for *text-classification*. We did not do automatic search for it, but manually tested, and results were already better on 3 epochs, and started *overfitting* on 4. We only tried a learning rate of *5e-5* as it was already better than *3e-5* during *MLM* experiment. And, for the batch size we used 8 with a *gradient accumulation steps* of 4 to simulate a batch size of 32 because of resource limitation.

2 Training parameters

For the training parameters, we used a batch size of 16 with a *gradient accumulation steps* of 4, a learning rate of *5e-5* as it was seen better during *MLM*, and epochs of 25, but the model performance was not getting better after 18 or 19 epochs depending on the model.

Experiments

In this section, we present you with the sample datasets that we used during the experiment and evaluation of the model.

1 BoW and TF-IDF dataset sample

Taubenflug und Wetter Mit dem geplanten Versuch möchten wir den Einfluss verschiedener Wetterbedingungen (Windrichtung und ?geschwindigkeit, Temperatur, Bewölkung/ Nebel) auf das Flugverhalten und Heimfindevermögen von Brieftauben untersuchen. Die Tiere werden hierzu mit hochauflösenden Biologgern ausgerüstet, die sowohl GPS-und Beschleunigungsloggern, als auch Magnetometer, Drucksensor, Temperatursensor, Gyroskop und Windgeschwindigkeitsmesser (Pitot Röhrchen) in Mikro-Ausführung beinhalten. Hiervon erwarten wir uns grundlegend neue Erkenntnisse zum Orientierungsverhalten von Vögeln, insbesondere von der Rolle visueller Landmarken. Wir erwarten keine Schäden für die Vögel. Die Tauben werden genau wie ""übliche"" Brieftauben eingesetzt, wobei die Heimkehrflüge deutlich unterhalb der Leistungsansprüche bleiben, die heute in Brieftaubenwettbewerben den Vögeln abverlangt werden. Einziger Unterschied zu den normalen Brieftauben ist die kleine gewichtsmäßige Zusatzbelastung durch die Messgeräte. Das Mehrgewicht liegt unterhalb 5% des Körpergewichtes der Taube und beeinträchtigt nach bisherigen Erfahrungen das Flugvermögen der Vögel nicht. Da die tauben bereits ab Schlupf die Handhabung durch eine Bezugsperson gewohnt sind, erleiden sie durch diesen Umgang auch keinen Stress. Die Beobachtungen des Orientierungs- und Flugverhaltens für diese Studie kann nur an ganzen und intakten Organismen, die sich in ihrer natürlichen Umwelt bewegen, erfolgen. Die Studie erfolgt bereits an der Tierart, die nach unserem Wissen durch Züchtung am besten an derartige Versuchsabläufe angepasst ist. Je 50 Tauben über einen Zeitraum von 5 Jahren stellen die nach unserer Erfahrung mit Brieftaubenauflässen geringst mögliche Zahl an Individuen dar, aus der sich aussagekräftige Ergebnisse erwarten lassen. Die Tauben erfahren ein optimales Training zur Vorbereitung der Heimkehrflüge und werden in bestmöglicher Kondition gehalten. Zum Einsatz kommende Messgergäte unterliegen weiterhin einer fortschreitenden Miniaturisierung, die wir genau verfolgen und kleinere und leichtere Loggertypen als diejenigen, die gegenwärtig technisch relaisierbar sind, werden umgehend zur Anwendung kommen.

Einfluss von Wachstumsfaktoren auf inhibitorischen Nervenzellen Neuronale Netzwerke bilden die funktionalen Einheiten unseres Gehirns. Komplexe Vorgänge wie Lernen und Erinnern beruhen auf dem Zusammenspiel zahlreicher, zum Teil unterschiedlicher Netzwerke in verschiedenen Regionen des Gehirns.Diese Netzwerke bestehen aus anregenden und inhibitorischen Nervenzellen. Inhibitorische Nervenzellen synthetisieren den hemmenden Neurotransmitter ?-Aminobuttersäure (GABA), daher werden diese Nervenzellen auch als GABAerge Nervenzellen bezeichnet. Die Entwicklung der einzelnen Klassen GABAerger Zellen wird durch genetischen Programme bestimmt, die von der Entstehungsregion der Zellen im frühembryonalen Gehirn abhängen. Aber weitgehend ungeklärt ist die Frage, welchen Einfluss sogenannte Wachstumsfaktoren auf die Entwicklung dieser Nervenzellen haben. Daher wollen wir nur in inhibitorischen Nervenzellen die Signalübertragung für neuronale Wachstumsfaktoren ausschalten und untersuchen, welche Folgen dies für die Entwicklung inhibitorischer Zellen und ihrer neuronaler Netzwerke haben. Solche komplexen Interaktionen können nicht in einem in-vitro Zellkultursystem untersucht werden. Das übergeordnete Ziel dieser Arbeit ist es, Einblicke in die Entwicklung inhibiorischer Nervenzellen sowie die Entwicklung der Kommunikation dieser Zellen mit anderen Nervenzellen zu bekommen. Diese Ergebnisse werden dazu beitragen, die neuronale Kommunikation zu entschlüsseln, die erst komplexe Lern- und Gedächtnissvorgänge ermöglicht. Nur im Gehirn von Mäusen wird die Signalübertragung für einen neuronalen Wachstumsfaktor nur in inhibitorischen Nervenzellen ausgeschaltet. Die dadurch versursachte Störung in der Inhibition des Gehirns verursacht Epilepsie in jungen Mäusen. Daher werden die Tiere schmerzfrei getötet und die Gehirne zur experimentellen Untersuchung entnommen, bevor die Lebensfunktionen der Mäuse durch die Epilepsie zu stark beeinträchtigt werden. Es wird ein mittlerer Schweregrad des Versuchsvorhabens erwartet. Der Aufbau des Gehirns sowie die Vernetzung der unterschiedlichen Nervenzellen lässt sich nur äußerst bedingt in in-vitro Zellkultursystemen nachvollziehen. Zudem entsprechen Zellkulturen oft einem nicht definierten Entwicklungsstand der betreffenden Zellen. Um Aussagen über die Entwicklung von unterschiedlichen Nervenzellen und Netzwerken im Gehirn treffen zu können, müssen wir auch im Gehirn unsere Untersuchungen durchführen. Die geschätzte Anzahl der Tiere basiert auf unseren derzeitigen Erfahrungen auf diesem Gebiet der Neurowissenschaften.Wir werden von einem Biostatistiker bei der Durchführung der Untersuchungen beraten, um sicherzustellen, dass die minimale Anzahl von Tieren verwendet wird, um das gewünschte Ergebnis zu erzielen. Die Mäuse mit gestörter Signalübertragung von Wachstumsfaktoren verbleiben mit ihren Geschwistertieren bei der Mutter. Die Mäuse sind in einer geschützten Umgebung untergebracht und mit Einstreu und Nistmaterial versorgt und um eine artgerechte Haltung zu gewährleisten. Außerdem wird das Befinden der Tiere regelmäßig überprüft. Falls ein Tier eine Infektion erleidet oder ernsthaft erkrankt, wird es vorzeitig schmerzfrei getötet.

STORM Präsynapse In diesem Versuchsvorhaben soll die molekulare Nanostruktur der präsynaptischen Kompartimente dreier verschiedener Synapsentypen des zentralen Nervensystems mittels STORM super-resolution Mikroskopie beschrieben werden. Die genaue Kenntnis der relativen Anordung verschiedener Proteine, Proteinkomplexe und funktioneller Einheiten innerhalb der Präsynapse ist essentiell für das Verständnis der Strukturen die den unterschiedlichen funktionellen Eigenschaften der zu untersuchenden Synapsen zugrunde liegen. Die beschriebenen Experimente werden an fixierten Hirnschnitten durchgeführt um einen möglichst physiologischen Kontext zu gewährleisten und die Übertragung der gewonnenen Erkenntnisse auf andere Synapsen, nicht zuletzt menschliche, zu ermöglichen. Wir erwarten von den generierten Ergebnisse einen tieferen Einblick in die molekulare Struktur der Präsynapse und damit der Mechismen der synaptischen Transmission was zu neuen Ansätzen zum Verständnis von Erkrankungen des zentralen Nervensystems führen kann. Es wird eine Injektion ins zentrale Nervensystem vorgenommen. Dazu ist ein operativer Eingriff unter Vollnarkose notwendig. Postoperative Schmerzen werden behandelt. Die hier untersuchten Synapsen, die Schaffer-Kollateralen, corticothalamische Riesensynapsen und die Held'sche Calyx, stellen ein breites funktionelles und morphologisches Spektrum zentraler Synapsen dar, so dass aus ihrer molekularen Nanostruktur in möglichst physiologischer Umgebung, weitreichende Rückschlüsse auf die Funktionen von verschiedenen Synapsentypen möglich sind. Diese speziellen Synapsen entwickeln sich allerdings nur im lebenden Tier und nicht in neuronalen Zellkulturen, da diese lediglich aus embryonalem Gewebe gewonnen werden können, wobei die Umgebung der Synapsen zerstört wird, was zu einem Verlust von Signalen führt, die für die Indentitätsbildung der Synapse und deren korrekte Reifung wichtig sind. Die Untersuchung an Säugersynapsen ist notwendig um Rückschlüsse auf das menschliche Gehirn zu erlauben. Die Anzahl beantragter Tiere basiert auf unseren langjährigen Erfahrungen im Bereich der synaptischen Transmissionsforschung und der Mikroskopie und ist auf das Minimum zur Erlangung belastbarer Daten reduziert. Alle Eingriffe werden unter Vollnarkose durchgeführt und die Schmerzen die den Tieren durch die Operation entstehen werden durch adäquate Behandlung auf ein Minimum reduziert.

Wt1 und Flossenregeneration beim Zebrafisch Adulten Säugetieren, inklusive dem Menschen, fehlt die Fähigkeit Schäden ankomplexen Organen wie dem Herz, der Nieren oder dem zentralen Nervensystem reparieren zu können. Schäden an diesen Organen stellen somit große medizinische Probleme dar. Im Gegensatz zu Säugern haben andere Wirbeltierklassen, wie z.B. Fische robuste Regenerationsfähigkeiten und können die Funktion vieler komplexer Organe nach Schädigung wiederherstellen, wie z.B. Herz, Niere und Flosse. Die Flossenregeneration stellt dabei ein einfaches experimentelles System dar, um Prinzpien und grundlegende Mechanismen der Regeneration zu verstehen. Das Gen wt1 wird während der Regeneration exprimiert, seine Funktion während

der Regeneration ist hingegen noch nicht verstanden. Ähnlichkeit und evolutionäre Konservierung von genetischenProgrammen macht eine Übertragbarkeit der Ergebnisse bezüglich wt1 und Regeneration von einer Spezies auf die andere, hier von Fisch auf Mensch, sehrwahrscheinlich. Es ist somit zu erwarten, dass die Aufklärung der Mechanismender Regenerationsprozesse im Fisch eine hohe Relevanz für die Untersuchungund Entwicklung therapeutischer Maßnahmen bei Organschäden beimMenschen haben wird. Die Ampuation der Flossespitze an Fischlarven wird unter Anästhesie durchgeführt und die Tiere danach regelmäßig kontrolliert.Am Ende des Versuches werden die Tiere schmerzfrei getötet. Die zuerwartenden Schmerzen und Belastungen für die Tiere sind insgesamt mäßigeinzuschätzen. Organregeneration ist ein komplexer Vorgang und die Untersuchung derbeteiligten Prozesse erfordert Studien im Gewebekontext. Im Zellkultursystemwerden Zellen aus ihrem räumlichen und zeitlichen Kontext isoliert und könnensomit nur Teile der hier zu analysierenden Aspekte aufzeigen. Eine Vermeidungdes beantragten Tierversuches ist somit nicht möglich. Die Versuchstierzahlen sind auf ein Minimum beschränkt. Trotzdem sind sieausreichend geplant um wissenschaftlich signifikante Daten zu erhalten. ProTier und Teilversuch können mehrere Analysen durchgeführt werden. DieGesamttierzahl wird somit möglichst gering gehalten. Es werden Flossenregenerationsexperimente an Fischlarven durchgeführt. Alle Eingriffe finden unter Narkose statt und das verwendete Anästhetikum hat analgetisches Potential. Die Tiere werden nach dem Eingriff regelmäßig kontrolliert. Unerwartetes Leiden wird dabei nach konkret benannten Abbruchkriterien bewertet und die Tiere gegebenenfalls schmerzfrei getötet um ein Leiden zu verhindern.

Wirksamkeit von Substanzen gegen Kälberdurchfall In der Studie erfolgen Untersuchungen zur Behandlung von parasitär bedingtem Kälberdurchfall mit verschiedenen Substanzen hinsichtlich der Wirksamkeit gegen den ursächlichen Parasiten. Viele Milchviehbetriebe sind mit diesem Erreger durchseucht, wobei es zu hohen ökonomischen Verlusten und zur massiven Beeinträchtigung des Tierwohls kommen kann. Mit dieser Studie können die zu testenden Substanzen hinsichtlich ihrer Eignung als Tierarzneimittel beurteilt werden, was durch bisher eingeschränkte, kausale Behandlungsmöglichkeiten zu einer erheblichen Verbesserung der Situation in den betroffenen Betrieben führen würde. Um eine Wirksamkeit zu testen muss bei allen Tieren Durchfall ausgelöst werden, der je nach tatsächlicher Wirksamkeit der Substanzen zu einer geringen bis mäßigen Belastung führt.Die Tiere der Behandlungsgruppe müssen aus Gründen der Lebensmittelsicherheit (nicht zugelassener Wirkstoff) nach der Studie getötet werden.Während der Studie treten nur geringe Belastungen auf. Die Dauer beträgt 21 Tage. Eine Wirksamkeit der zu testenden Substanzen wurde in vitro und im Nagermodell bereits nachgewiesen.Für den Parasiten stellt die Maus jedoch einen Fehlwirt dar, in dem er sich nur sehr schwer vermehren kann. Daher muss hier mit immunsupprimierten Tieren gearbeitet werden, was einen Vergleich mit dem Kalb aufgrund der stark abweichenden Physi-

ologie und Immunologie unmöglich macht. Um eine Wirksamkeit im natürlichen Wirt des Parasiten zu zeigen und mit dem Hintergrund auf Grundlage der zu testenden Moleküle ein Tierarzneimittel zu entwickeln, ist die Testung in der Zieltierart in vivo laut Arzneimittelgesetz essentiell. Die Versuchstiere sind neugeborene Kälber. Sie erkranken häufig an Pneumonien oder Nabelentzündung (typische Kälber-krankheiten), müssen dann behandelt und in Folge dessen gegebenenfalls aus der Studie ausgeschlossen werden. Aus unseren Erfahrungen mit Tierversuchen, die auf einem ähnlichen Studiendesign basierten, planen wir von vornherein 9 Tiere pro Gruppe ein, um Fehlschläge zu vermeiden, die eine unnötige Belastung der Versuchstiere und ggf. durch Wiederholung höhere Tierzahlen bedeuten würden. Es wird nur ein leichtes bis moderates Durchfallgeschehen induziert. Treten Dehydration und Nebenerkrankungen auf, werden diese umgehend behandelt.Das Kalb ist die Zieltierart für den Kälberdurchfall verursachenden Erreger. Mögliche folgende klinischen Studien zur Zulassung eines Tierarzeimittels würden auf den Daten basieren, die durch diese Pilotstudie erhoben werden. Klinische Studien müssen immer in der Zieltierart durchgeführt werden.

2 Masking experiment sample dataset

2.1 NTS dataset

Sentence dataset

ER-Analysen bei neurodegenerativen Erkrankungen
Bereits von uns hergestellte Mausmodelle für neurodegenerative Erkrankungen sollen mit einer Reporter-Maus verpaart werden, Inder das ER mit einem fluoreszierenden Protein markiert ist.
Mit diesem Ansatz soll überprüft werden, ob den Erkrankungen Veränderungen der ER-Struktur zugrunde liegen.
Alle Experimente werden standardisiert durchgeführt, so dass dieGruppengröße für Experimente relativ klein ist.
Dieselben Tiere werden soweit möglich mehrfach verwendet.
Sollten aufgrund unvorhergesehener Befunde Änderungen im Versuchsablaufnotwendig sein, werden wir eine entsprechende Änderung beantragen.
Inhibition von Meprin in assoziierten Krankheiten
Es wurde kürzlich gezeigt, dass Meprine unter den Proteasen eine wichtige Rolle im Organismus spielen.
Die multiple sklerose (MS) ist eine autoimmune Erkrankung des zentralen Nervensystems.

Article dataset

Signaltransduktion bei der akuten Hepatitis B VirusinfektionDie chronische Hepatitis B Virusinfektion (HBV) ist ein wesentlicher Risikofaktor für die Entstehung einer Leberzirrhose und ihrer Komplikationen und oft schwer zu therapieren. Der Transkriptionsfaktor AP-1 und purinerge Rezeptoren spielen eine wesentliche Rolle bei der Signaltransduktion im Rahmen akuter Leberentzündungen und könnten wesentliche Funktionen vom Übergang einer akuten zur chronischen Hepatitis spielen und so neue Therapieansätze für Virushepatitiden eröffnen. Vor diesem Hintergrund sollen die Funktionen dieser Signaltransduktionswege in einem Mausmodell einer akuten HBV-Infektion genauer charakterisiert werden.Im Rahmen dieses Projektes kommen etablierte Mausmodelle zum Einsatz, die in Fachkreisen routinemäßig für Studien der akuten HBV Infektion verwendet werden: Isolation primärer Hepatozyten: geringe Belastung. Modulation der Genexpression durch Injektion von RNA in MxCre-transgenenMäusgeringe Beeinträchtigung;Akute Hepatitis durch Infektion mit adenoviralen Vektoren: geringe bis mäßige Belastung.Alle Tiere werden am Versuchsende zur Organentnahme getötet.Bei akuten Virushepatitiden handelt es sich um komplexe systemische Erkrankungen, die sich in Zellkulturmodellen nicht hinreichend abbilden lassen. Zudem stehen keine spezifischen und effizienten Methoden zur Verfügung, um die Expression von AP-1 sowie P2 Rezeptoren zu hemmen. Daher stellt die Verwendung entsprechender Knockoutmausmodellen ein alternativloses in vivo System dar, um die Funktionen dieser Signaltransduktionswege bei der akuten HBV Infektion zu charakterisieren.Die Zahl der zu verwendenden Tiere wurde auf das unerläßliche Maß reduziert. Durch ein zweistufiges Verfahren sollen zunächst in Pilotexperimenten die Funktionen von 5 Genen bei der akuten HBV Infektion überprüft und im 2. Teil nur die beiden Gene mit den eindeutigsten Phänotypen genauer charakterisiert werden.Es werden durch die vewendeten Versuchsprotokolle keine höhergradigen Beeinträchtigungen der Tiere erwartet. Andernfalls erfolgt in Rücksprache mit dem Tierschutzbeauftragten eine analgetische Therapie bzw. der Versuchsabbruch. Es werden etablierte Mausmodelle der akuten HBV-Infektion verwendet. Die Infektion von Mäusen mit adenoviralen Vektoren ist erforderlich, da sich die akute Infektion mit humanem HBV sonst nur in Schimpansen oder Tupaia belangeri untersuchen lässt.

mtDNA und Immunzellen in Entzündungen der HautBlasenbildende Autoimmunerkrankungen der Haut sind schwerwiegende Erkrankungen, die unbehandelt tödlich verlaufen können. Auch wenn viele Fortschritte bei der Untersuchung der Pathogenese dieser Erkrankungen gemacht wurden, sind viele Einflussgrößen, welche den Verlauf dieser Erkrankungen modulieren weitgehend noch nicht untersucht oder unbekannt. In diesem Versuchsvorhaben sollen zwei bislang kaum untersuchte Immunzellpopulationen auf deren Funktion in einer blasenbildende Autoimmunerkrankung untersucht werden. In diesem Zusammenhang soll weiterhin eine genetischen Variation näher untersucht werden, zu welcher bereits bekannt ist, dass diese das metabolische Gleichgewicht

verändert und den Verlauf der Erkrankung deutlich abmildert. Dieses Versuchsvorhaben soll dazu beitragen ein besseres Verständnis über die Interaktionen von Immunzellen und Entzündungsreaktionen in dieser Erkrankung zu erlangen und zu verstehen, auf welche Weise genetische Variationen mit Wirkung auf den Metabolismus ebenfalls auf das Immunsystem einwirken.Die Erkenntnisse die in diesem Versuchsvorhaben gewonnen werden sollen, dienen der Entwicklung neuer therapeutischer Behandlungsmethoden und therapiebegleitenden Maßnahmen für blasenbildende Autoimmunerkrankungen, sowie als Grundansatz für andere Autoimmunerkrankungen.wiederholten Eingriffe (Applikation von Substanzen, klinische Untersuchung inNarkose) führen bei den Tieren zu einer insgesamt mäßigen Belastung, die übereinen Zeitraum von 12 Tagen andauert. Am Versuchsende erleiden die Tiere diemaximale Belastung, da sie getötet werden, um die weiterführende Untersuchungen an Haut und anderen Organen zu ermöglichen.Die komplexen Interaktionen, die zu einer kutanen Entzündung führen könnenleider nicht in vitro nachgestellt werden.Mithilfe von statistischen Planungsverfahren wurde die minimale Tieranzahlberechnet, die für diesen Versuch notwendig ist.Die Modelle der hier zu untersuchenden Hautentzündung wurden bisher nur inMäusen etabliert. In anderen Tieren gibt es keine Modelle.

2.2 Wikipedia dataset

Sentence dataset

Der Ort entstand vermutlich im Verlauf der Reconquista .
Vor allem wenn man sieht , dass die anderen hinzugefügten Links zwar nichts mit der visuellen Kontrolle aber doch auch wieder mit Russland zu tun haben .
Die notwendige Infrastruktur wurde dort 1831 – 1835 fertiggestellt .
Papst Johannes Paul II . sprach am 3. Oktober 1998 in Marija Bistrica Kardinal Stepinac selig .
Das Männlein dort auf einem Bein
Ende 1923 verließ er England in Richtung Amerika .
Zu dieser Zeit bestand Žalkovice aus 110 Häusern und hatte etwa 700 Einwohner .
Siehe Auswärtiges Amt , diverse Atlanten (auch Schulatlanten) und der Duden .

Articles dataset

Die Besteuerung der Mineralienexporte stellte die hauptsächliche Finanzierung des Krieges dar , obwohl zahlreiche Versuche , den Handel mit Bergbauprodukten zu monopolisieren , immer wieder fehlschlugen . Nach deren Abschluss befand sich die " Falke " vom 2. bis zum 20. Oktober für Probefahrten im Dienst . Da die FIFA die Spiele als Freundschaftsspiele einstufte , durfte jede Mannschaft pro Spiel sechs Spielerinnen auswechseln , in maximal vier Spielunterbrechungen , davon maximal drei

Spielunterbrechungen nach Beginn der zweiten Halbzeit . Bei den Herren gewann die Mannschaft 1995 , 2007 und 2011 die Bronze- , sowie 1999 und 2003 die Silbermedaille . Er war bis 1976 in Leuna verantwortlich für die technologische Entwicklung und Weiterentwicklung des Hochdruckpolyethylenverfahrens , die ab 1969 in Zusammenarbeit mit der Sowjetunion zum Aufbau der 50 kt / a Anlage " Polymir 50 " in Nowopolozk in Weißrussland und der 60 kt / a Anlage " Poymir 60 " in Leuna führte . 1976 – 1980 war er Abteilungsleiter der Hochdruckpolyethylensynthese , 1981 – 1984 Forschungsdirektor und 1985 – 1990 Produktionsdirektor in den Leunawerken . Die Käfer werden 12 bis 16 Millimeter lang . Eckhardt war Mitbegründer und kultureller Leiter der österreichisch-amerikanischen Gesellschaft . Dass das römische Gräberfeld , auf dem St. Ursula errichtet wurde , bereits im 12. Jahrhundert auf der Suche nach Reliquien stark durchwühlt wurde , erschwerte die archäologischen Auswertungen der 1940er Jahre – exakte Klarheit über die Veränderungen an dem Bau lässt sich wohl nicht mehr erzielen . Labine ist seit 2008 mit der Schauspielerin Carrie Ruscheinsky verheiratet . Republiken zu- und gleichzeitig deren eigene Bewegungsfreiheit abnimmt . " Von beiden Dreiecken braucht nur noch das gemeinsame Dreieck formula_5 abgezogen zu werden . Als erster Premierminister gilt Robert Walpole (1721 – 1742) , derzeitige Amtsinhaberin ist Theresa May . Am 30. Januar 1944 wurde sie ins KZ Auschwitz-Birkenau deportiert , das sie sieben Tage später , am 6. Februar 1944 , erreichte . Henning wurde 1873 infolge des Baus einer Eisenbahnstation gegründet und zwei Jahre später amtlich anerkannt . Die Liste der Bahnhöfe und Haltepunkte in Sachsen führt alle Eisenbahnbetriebsstellen des öffentlichen Verkehrs auf , die auf dem Territorium des 1990 neu begründeten Freistaats Sachsen bestehen . Die „ Roten Raketen “ aus Berlin (später „ Sturmtrupp Alarm “) beispielsweise tourten als Werbetruppe des RFB mit einem alten Auto durch das Land . Er ist (was in der englischen WP-Ausgabe leider öfter als in der deutschen vorkommt) voller Redundanzen und Geschwurbel . Die befindet sich im Besitz der neuseeländischen , die im Januar 2015 durch Umbenennung aus der " hervorgegangen ist . Die Chromosomenzahl ist 2n = 16. " On the distribution of prime numbers .

Irgendwann übernimmt es jemand in einen Zeitungsartikel oder gar in ein Buch , und schon haben wir sogar (nachträglich) einen Beleg für den Wikipdia-Beitrag . Endgültig verraten glaubt sich Christian , als sein einziger Freund Robert mit seiner Schwester Helga anbandelt . Als ihr klar wird , dass sie wieder gegangen sind , will sie ihren gegenüber Matthew angekündigten Selbstmord für den Fall , dass ihre Eltern die Geschwisterliebe bemerken , verwirklichen . Die Larven sind ebenfalls gelb und haben am ganzen Körper schwarze Fortsätze , aus denen Büschel mit schwarzen Haaren wachsen . Fahnenträger bei der Eröffnungsfeier war Nader Masri . Die Art wurde als " Himantura chaophraya " 1990 beschrieben , jedoch stellte sich im Jahr 2008 bei Vergleichen durch Last und Manjaji-Matsumoto heraus , dass die Art mit dem Typusmaterial von " H. polylepis " übereinstimmt . Der König ließ daraufhin die Kirche mit seinem Grab von einer bewaffneten

Wache absperren . 1323 kam es zu Unruhen am Ort seiner Hinrichtung , wo sich Menschenmassen zum Gebet versammelten . Dort kam u. a. der damalige Verband des Sonderkommandos Elbe zum Einsatz . Im März 2011 wurde Michajlow im Exekutivkomitee der UEFA gewählt . Die Umkehrzeit ist ebenfalls sehr wichtig : Die Bergsteiger sollten unter allen Umständen bis zum Einbruch der Dunkelheit am Gipfeltag das Hochlager wieder erreicht haben . Alle Gebäude sind scheinbar nach Plänen des " Seigneur Constance " , wie Phaulkon von den Franzosen genannt wurde , in einem europäischen Stil erbaut , die Wohnhäuser sind zweistöckig ausgeführt . Doppelt vorhanden waren auch die Baugruppen Telemetriekontrolle , Signalverarbeitung und Transponder . Schon 1984 gab Stein seine Professur in Wien wieder auf , um auf Bitte von Landesbischof Klaus Engelhardt das Amt des geschäftsleitenden Oberkirchenrates im Evangelischen Oberkirchenrat der Evangelischen Landeskirche in Baden zu übernehmen . Richard Parkinson (* 1844 in Augustenborg auf der Insel Alsen , Dänemark ; † 1909) war ein deutscher Südseeforscher und Kolonist . Das den Behälter umschließende Mauerwerk ragt deutlich vor und ist in Fachwerkbauweise mit Eisenfachung ausgeführt . Die erste Version erschien 1963 , 1973 erfolgten die Unterschriften und am 1. Juli 1975 trat es in Kraft . Ein- oder zweimal im Jahr bringt das Weibchen nach rund 60-tägiger Tragzeit ein bis drei Jungtiere zur Welt . Man tappt weiterhin im Dunkel , sowohl über die Ursache als auch über die Therapie . Beim Strömungstauchen im Wildwasser können Sicherungsleinen über oder im Wasser sinnvoll sein . Dabei werden Details verändert .

3 Sub-Word Token prediction for NTS articles

Sentence: [CLS] Signal ##tra ##ns ##duktion bei der ak ##uten He ##pat ##itis B Virus ##in ##fektion ##Die chron ##ische He ##pat ##itis B Virus ##in ##fektion (H ##BV) ist ein wesentlicher Risiko ##faktor für die Entstehung einer Leber ##zir ##r ##hose und ihrer Kompl ##ikationen und oft schwer zu ther ##ap ##ieren [SEP] Der Trans ##k ##ript ##ions ##faktor AP - 1 und pur ##iner ##ge Rezept ##oren spielen eine wesentliche Rolle bei der Signal ##tra ##ns ##duktion im Rahmen ak ##uter Leber ##entz ##ündung ##en und könnten wesentliche Funktionen vom Übergang einer ak ##uten zur chron ##ischen He ##pat ##itis spielen und so neue Therapie ##ans ##ätze für Virus ##he ##pat ##iti ##den eröffnen [SEP] Vor diesem Hintergrund sollen die Funktionen dieser Signal ##tra ##ns ##duktion ##s ##wege in einem Maus ##modell einer ak ##uten H ##BV - Infektion genauer charakter ##isiert werden [SEP] Im Rahmen dieses Projekte ##s kommen etablierte Maus ##modelle zum Einsatz , die in Fach ##kreisen ro ##utin ##em ##äß ##ig für Studien der ak ##uten H ##BV Infektion verwendet werden : Isol ##ation primär ##er He ##pat ##ozy ##ten : geringe Belastung [SEP] Mod ##ulation der Gene ##x ##pression durch In ##jekt ##ion von R ##N ##A in M ##x ##C ##re - trans ##gen ##en ##M ##äus ##ger ##inge

Beeinträcht ##igung ; Ak ##ute He ##pat ##itis durch Infektion mit ad ##en ##ovi ##ralen Ve ##ktoren : geringe bis mäß ##ige Belastung [SEP] Alle Tiere werden am Versuchs ##ende zur Organe ##nt ##nahme getötet [SEP] Bei ak ##uten Virus ##he ##pat ##iti ##den handelt es sich um komplexe system ##ische Erkrankungen , die sich in Zell ##kultur ##modelle ##n nicht hinreichend ab ##bilden lassen [SEP] Zudem stehen keine spezifischen und effiz ##ienten Methoden zur Verfügung , um die Express ##ion von AP - 1 sowie P ##2 Rezept ##oren zu he ##mm ##en [SEP] Daher stellt die Verwendung entsprechender Kn ##ock ##out ##maus ##modelle ##n ein altern ##ativ ##lose ##s in v ##iv ##o System dar , um die Funktionen dieser Signal ##tra ##ns ##duktion ##s ##wege bei der ak ##uten H ##BV Infektion zu charakter ##isieren [SEP] Die Zahl der zu verwenden ##den Tiere wurde auf das uner ##lä ##ß ##liche Maß reduziert [SEP] Durch ein zwei ##stuf ##iges Verfahren sollen zunächst in Pilot ##ex ##per ##imente ##n die Funktionen von 5 Gene ##n bei der ak ##uten H ##BV Infektion überprüft und im 2 [SEP] Teil nur die beiden Gene mit den eindeutig ##sten Ph ##än ##otyp ##en genauer charakter ##isiert werden [SEP] Es werden durch die ve ##wendet ##en Versuchs ##protokoll ##e keine höher ##grad ##igen Beeinträcht ##igungen der Tiere erwartet [SEP] Andern ##falls erfolgt in Rück ##sprache mit dem Tierschutz ##be ##auftragten eine anal ##get ##ische Therapie bzw [SEP] der Versuchs ##ab ##bruch [SEP] Es werden etablierte Maus ##modelle der ak ##uten H ##BV - Infektion verwendet [SEP] Die Infektion von Mä ##usen mit ad ##en ##ovi ##ralen Ve ##ktoren ist erforderlich , da sich die ak ##ute Infektion mit human ##em H ##BV sonst nur in Sch ##imp

Masked Sentence: [CLS] Signal ##tra ##ns ##duktion bei der ak ##uten He ##pat ##itis B Virus [MASK] [MASK] ##Die chron [MASK] He ##pat ##itis B Virus ##in ##fektion (H ##BV) ist ein wesentlicher Risiko ##faktor für die Entstehung einer Leber ##zir ##r ##hose und ihrer Kompl ##ikationen und oft schwer zu ther ##ap ##ieren [SEP] Der Trans ##k ##ript ##ions ##faktor AP - 1 und pur ##iner ##ge Rezept ##oren spielen eine wesentliche Rolle bei der Signal ##tra ##ns ##duktion im Rahmen ak ##uter Leber ##entz ##ündung ##en und könnten wesentliche Funktionen vom Übergang einer ak ##uten zur chron ##ischen He ##pat ##itis spielen und so neue Therapie ##ans ##ätze für Virus ##he ##pat ##iti ##den eröffnen [SEP] Vor diesem Hintergrund sollen die Funktionen dieser Signal ##tra ##ns ##duktion ##s ##wege in einem Maus ##modell einer ak ##uten H ##BV - Infektion genauer charakter ##isiert werden [SEP] Im Rahmen dieses Projekte ##s kommen etablierte Maus ##modelle zum Einsatz , die in Fach ##kreisen ro ##utin ##em ##äß ##ig für Studien der ak ##uten H ##BV Infektion verwendet werden : Isol ##ation primär ##er He ##pat ##ozy ##ten : geringe Belastung [SEP] Mod ##ulation der Gene ##x ##pression durch In ##jekt ##ion von R ##N ##A in M ##x ##C ##re - trans ##gen ##en ##M ##äus ##ger ##inge Beeinträcht ##igung ; Ak ##ute He ##pat ##itis durch Infektion mit ad ##en ##ovi ##ralen Ve ##ktoren : geringe bis mäß ##ige Belas-

tung [SEP] Alle Tiere werden am Versuchs ##ende zur Organe ##nt ##nahme getötet [SEP] Bei ak ##uten Virus ##he ##pat ##iti ##den handelt es sich um komplexe system ##ische Erkrankungen , die sich in Zell ##kultur ##modelle ##n nicht hinreichend ab ##bilden lassen [SEP] Zudem stehen keine spezifischen und effiz ##ienten Methoden zur Verfügung , um die Express ##ion von AP - 1 sowie P ##2 Rezept ##oren zu he ##mm ##en [SEP] Daher stellt die Verwendung entsprechender Kn ##ock ##out ##maus ##modelle ##n ein altern ##ativ ##lose ##s in v ##iv ##o System dar , um die Funktionen dieser Signal ##tra ##ns ##duktion ##s ##wege bei der ak ##uten H ##BV Infektion zu charakter ##isieren [SEP] Die Zahl der zu verwenden ##den Tiere wurde auf das uner ##lä ##ß ##liche Maß reduziert [SEP] Durch ein zwei ##stuf ##iges Verfahren sollen zunächst in Pilot ##ex ##per ##imente ##n die Funktionen von 5 Gene ##n bei der ak ##uten H ##BV Infektion überprüft und im 2 [SEP] Teil nur die beiden Gene mit den eindeutig ##sten Ph ##än ##otyp ##en genauer charakter ##isiert werden [SEP] Es werden durch die ve ##wendet ##en Versuchs ##protokoll ##e keine höher ##grad ##igen Beeinträcht ##igungen der Tiere erwartet [SEP] Andern ##falls erfolgt in Rück ##sprache mit dem Tierschutz ##be ##auftragten eine anal ##get ##ische Therapie bzw [SEP] der Versuchs ##ab ##bruch [SEP] Es werden etablierte Maus ##modelle der ak ##uten H ##BV - Infektion verwendet [SEP] Die Infektion von Mä ##usen mit ad ##en ##ovi ##ralen Ve ##ktoren ist erforderlich , da sich die ak ##ute Infektion mit human ##em H ##BV sonst nur in Sch ##imp
Masked tokens: ['##in', '##fektion', '##ische']
Predicted tokens: ['[unused_punctuation0]', '##2', '##ische']

4 Whole Word Token prediction for NTS articles

Sentence: [CLS] Signal ##tra ##ns ##duktion bei der ak ##uten He ##pat ##itis B Virus ##in ##fektion ##Die chron ##ische He ##pat ##itis B Virus ##in ##fektion (H ##BV) ist ein wesentlicher Risiko ##faktor für die Entstehung einer Leber ##zir ##r ##hose und ihrer Kompl ##ikationen und oft schwer zu ther ##ap ##ieren [SEP] Der Trans ##k ##ript ##ions ##faktor AP - 1 und pur ##iner ##ge Rezept ##oren spielen eine wesentliche Rolle bei der Signal ##tra ##ns ##duktion im Rahmen ak ##uter Leber ##entz ##ündung ##en und könnten wesentliche Funktionen vom Übergang einer ak ##uten zur chron ##ischen He ##pat ##itis spielen und so neue Therapie ##ans ##ätze für Virus ##he ##pat ##iti ##den eröffnen [SEP] Vor diesem Hintergrund sollen die Funktionen dieser Signal ##tra ##ns ##duktion ##s ##wege in einem Maus ##modell einer ak ##uten H ##BV - Infektion genauer charakter ##isiert werden [SEP] Im Rahmen dieses Projekte ##s kommen etablierte Maus ##modelle zum Einsatz , die in Fach ##kreisen ro ##utin ##em ##äß ##ig für Studien der ak ##uten H ##BV Infektion verwendet werden : Isol ##ation primär ##er He ##pat ##ozy ##ten

: geringe Belastung [SEP] Mod ##ulation der Gene ##x ##pression durch In ##jekt ##ion von R ##N ##A in M ##x ##C ##re - trans ##gen ##en ##M ##äus ##ger ##inge Beeinträcht ##igung ; Ak ##ute He ##pat ##itis durch Infektion mit ad ##en ##ovi ##ralen Ve ##ktoren : geringe bis mäß ##ige Belastung [SEP] Alle Tiere werden am Versuchs ##ende zur Organe ##nt ##nahme getötet [SEP] Bei ak ##uten Virus ##he ##pat ##iti ##den handelt es sich um komplexe system ##ische Erkrankungen , die sich in Zell ##kultur ##modelle ##n nicht hinreichend ab ##bilden lassen [SEP] Zudem stehen keine spezifischen und effiz ##ienten Methoden zur Verfügung , um die Express ##ion von AP - 1 sowie P ##2 Rezept ##oren zu he ##mm ##en [SEP] Daher stellt die Verwendung entsprechender Kn ##ock ##out ##maus ##modelle ##n ein altern ##ativ ##lose ##s in v ##iv ##o System dar , um die Funktionen dieser Signal ##tra ##ns ##duktion ##s ##wege bei der ak ##uten H ##BV Infektion zu charakter ##isieren [SEP] Die Zahl der zu verwenden ##den Tiere wurde auf das uner ##lä ##ß ##liche Maß reduziert [SEP] Durch ein zwei ##stuf ##iges Verfahren sollen zunächst in Pilot ##ex ##per ##imente ##n die Funktionen von 5 Gene ##n bei der ak ##uten H ##BV Infektion überprüft und im 2 [SEP] Teil nur die beiden Gene mit den eindeutig ##sten Ph ##än ##otyp ##en genauer charakter ##isiert werden [SEP] Es werden durch die ve ##wendet ##en Versuchs ##protokoll ##e keine höher ##grad ##igen Beeinträcht ##igungen der Tiere erwartet [SEP] Andern ##falls erfolgt in Rück ##sprache mit dem Tierschutz ##be ##auftragten eine anal ##get ##ische Therapie bzw [SEP] der Versuchs ##ab ##bruch [SEP] Es werden etablierte Maus ##modelle der ak ##uten H ##BV - Infektion verwendet [SEP] Die Infektion von Mä ##usen mit ad ##en ##ovi ##ralen Ve ##ktoren ist erforderlich , da sich die ak ##ute Infektion mit human ##em H ##BV sonst nur in Sch ##imp

Masked Sentence: [CLS] Signal ##tra ##ns ##duktion bei der ak ##uten He ##pat ##itis B [MASK] [MASK] [MASK] ##Die chron ##ische He ##pat ##itis B Virus ##in ##fektion (H ##BV) ist ein wesentlicher Risiko ##faktor für die Entstehung einer Leber ##zir ##r ##hose und ihrer Kompl ##ikationen und oft schwer zu ther ##ap ##ieren [SEP] Der Trans ##k ##ript ##ions ##faktor AP - 1 und pur ##iner ##ge Rezept ##oren spielen eine wesentliche Rolle bei der Signal ##tra ##ns ##duktion im Rahmen ak ##uter Leber ##entz ##ündung ##en und könnten wesentliche Funktionen vom Übergang einer ak ##uten zur chron ##ischen He ##pat ##itis spielen und so neue Therapie ##ans ##ätze für Virus ##he ##pat ##iti ##den eröffnen [SEP] Vor diesem Hintergrund sollen die Funktionen dieser Signal ##tra ##ns ##duktion ##s ##wege in einem Maus ##modell einer ak ##uten H ##BV - Infektion genauer charakter ##isiert werden [SEP] Im Rahmen dieses Projekte ##s kommen etablierte Maus ##modelle zum Einsatz , die in Fach ##kreisen ro ##utin ##em ##äß ##ig für Studien der ak ##uten H ##BV Infektion verwendet werden : Isol ##ation primär ##er He ##pat ##ozy ##ten : geringe Belastung [SEP] Mod ##ulation der Gene ##x ##pression durch In ##jekt ##ion von R ##N ##A in M ##x ##C ##re - trans ##gen

##en ##M ##äus ##ger ##inge Beeinträcht ##igung ; Ak ##ute He ##pat ##itis durch Infektion mit ad ##en ##ovi ##ralen Ve ##ktoren : geringe bis mäß ##ige Belastung [SEP] Alle Tiere werden am Versuchs ##ende zur Organe ##nt ##nahme getötet [SEP] Bei ak ##uten Virus ##he ##pat ##iti ##den handelt es sich um komplexe system ##ische Erkrankungen , die sich in Zell ##kultur ##modelle ##n nicht hinreichend ab ##bilden lassen [SEP] Zudem stehen keine spezifischen und effiz ##ienten Methoden zur Verfügung , um die Express ##ion von AP - 1 sowie P ##2 Rezept ##oren zu he ##mm ##en [SEP] Daher stellt die Verwendung entsprechender Kn ##ock ##out ##maus ##modelle ##n ein altern ##ativ ##lose ##s in v ##iv ##o System dar , um die Funktionen dieser Signal ##tra ##ns ##duktion ##s ##wege bei der ak ##uten H ##BV Infektion zu charakter ##isieren [SEP] Die Zahl der zu verwenden ##den Tiere wurde auf das uner ##lä ##ß ##liche Maß reduziert [SEP] Durch ein zwei ##stuf ##iges Verfahren sollen zunächst in Pilot ##ex ##per ##imente ##n die Funktionen von 5 Gene ##n bei der ak ##uten H ##BV Infektion überprüft und im 2 [SEP] Teil nur die beiden Gene mit den eindeutig ##sten Ph ##än ##otyp ##en genauer charakter ##isiert werden [SEP] Es werden durch die ve ##wendet ##en Versuchs ##protokoll ##e keine höher ##grad ##igen Beeinträcht ##igungen der Tiere erwartet [SEP] Andern ##falls erfolgt in Rück ##sprache mit dem Tierschutz ##be ##auftragten eine anal ##get ##ische Therapie bzw [SEP] der Versuchs ##ab ##bruch [SEP] Es werden etablierte Maus ##modelle der ak ##uten H ##BV - Infektion verwendet [SEP] Die Infektion von Mä ##usen mit ad ##en ##ovi ##ralen Ve ##ktoren ist erforderlich , da sich die ak ##ute Infektion mit human ##em H ##BV sonst nur in Sch ##imp
Masked tokens: ['Virus', '##in', '##fektion']
Predicted tokens: ['[unused_punctuation0]', '##2', '##ische']

5 Sanity check on NTS token predictions

5.1 NTS-random words sentences

Signaltransduktion bei der akuten Hepatitis B VirusÄnderung
Die chronische Infektion mit dem B-Virus (HBV) ist ein wichtiger Risikofaktor für die Verbesserung der Leberzirrhose und ihre Besserung und ist oft gut zu behandeln.
Der Transkriptionsfaktor AP-1 und purinerge Faktoren spielen eine wesentliche Rolle bei der Signaltransduktion im Rahmen der akuten Leberkühlung und könnten wesentliche Funktionen beim Übergang vom akuten zum chronischen Fieber spielen und damit einen neuen therapeutischen Versuch für die Virushepatitis eröffnen.
Im Gegensatz zu dieser Geschichte werden die Dinge dieser Signalflusswege in einem Mausmodell einer falschen HBV-Infektion anders aussehen.
In diesem Projekt werden wütende Mausmodelle verwendet, die in Fachkreisen immer wieder für das Spiel der akuten HBV-Infektion eingesetzt werden: Gruppe von primären Hepatozyten: geringe Belastung.

Modulation des Genverhaltens durch Injektion von Saft in transgene MxCre-Mäuse Beeinträchtigung;Akute Hepatitis durch Überdosierung mit adenoviralen Vektoren: geringe bis mittlere Exposition.
Alle Tiere werden am Ende des Spiels zur Organentnahme versprochen.
Die akute Virushepatitis ist ein einfaches systemisches Spiel, das in Zellkulturmodellen nicht ausreichend p sein kann.
Darüber hinaus sind keine spezifischen und wirksamen Medikamente zur Hemmung der Expression von AP-1- und P2-Rezeptoren auf Lager.
Daher bietet die Verwendung von starken Knockout-Mausmodellen ein festes in vivo-System, um die Funktionen dieser Signaltransduktionswege bei einer akuten HBV-Infektion zu aktivieren.

5.2 NTS-original sentences

Signaltransduktion bei der akuten Hepatitis B Virusinfektion
Die chronische Hepatitis B Virusinfektion (HBV) ist ein wesentlicher Risikofaktor für die Entstehung einer Leberzirrhose und ihrer Komplikationen und oft schwer zu therapieren.
Der Transkriptionsfaktor AP-1 und purinerge Rezeptoren spielen eine wesentliche Rolle bei der Signaltransduktion im Rahmen akuter Leberentzündungen und könnten wesentliche Funktionen vom Übergang einer akuten zur chronischen Hepatitis spielen und so neue Therapieansätze für Virushepatitiden eröffnen.
Vor diesem Hintergrund sollen die Funktionen dieser Signaltransduktionswege in einem Mausmodell einer akuten HBV-Infektion genauer charakterisiert werden.
Im Rahmen dieses Projektes kommen etablierte Mausmodelle zum Einsatz, die in Fachkreisen routinemäßig für Studien der akuten HBV Infektion verwendet werden: Isolation primärer Hepatozyten: geringe Belastung.
Modulation der Genexpression durch Injektion von RNA in MxCre-transgenenMäusgeringe Beeinträchtigung;Akute Hepatitis durch Infektion mit adenoviralen Vektoren: geringe bis mäßige Belastung.
Alle Tiere werden am Versuchsende zur Organentnahme getötet.
Bei akuten Virushepatitiden handelt es sich um komplexe systemische Erkrankungen, die sich in Zellkulturmodellen nicht hinreichend abbilden lassen.
Zudem stehen keine spezifischen und effizienten Methoden zur Verfügung, um die Expression von AP-1 sowie P2 Rezeptoren zu hemmen.
Daher stellt die Verwendung entsprechender Knockoutmausmodellen ein alternativloses in vivo System dar, um die Funktionen dieser Signaltransduktionswege bei der akuten HBV Infektion zu charakterisieren.

6 NTS datasets to evaluate model

6.1 Doc 6198

Einfluss von Crlf1 auf die Synthese und Proteolyse von Proteinen
Das hier vorgestellte Versuchsvorhaben dient der medizinischen Grundlagenforschung mit unmittelbarem klinischen Praxisbezug. Der Tierversuch zielt auf die Entwicklung einer Therapie zur Behandlung des **Crisponi Syndroms** ab. Das **Crisponi Syndrom** ist eine seltene autosomal-rezessive Erkrankung, die aufgrund von Hyperthermien häufig schon im Säuglingsalter zum Tod der Patienten führt. Einige wenige Patienten, die überleben, zeigen in der Adolezenz ein paradoxes Schwitzen bei Temperaturen unter 20C. Therapeutische Interventionen sind bisher auf unterstützende Maßnahmen wie die Behandlung des paradoxen Schwitzens begrenzt. Trotz der guten phänotypischen Charakterisierung des Crisponi Syndroms steht leider bisher keine spezifische Behandlung zur Verfügung. Daher ist die Erforschung der Pathomechanismen des Crisponi Syndroms dringend erforderlich. Das Hauptziel dieses Forschungsprojekts ist, den Pathomechanismus dieser seltenen Erkrankung an neuronalen Zellen einer Crlf1-Knockoutmaus weiter aufzuklären. Diese Zellen werden auf die Expression noradrenerger Gene und eine veränderte Proteolyse im Vergleich zu Wildtypzellen hin überprüft. Wir erwarten durch dieses Projekt die Grundlage für die Entwicklung neuer Therapieansätze zu schaffen.
Im Rahmen der vorgesehenen Behandlung sind erhebliche, länger anhaltende oder sich wiederholende Schmerzen nicht zu erwarten. Erwartete Belastung: gering.
Wir wollen den Pathomechanismus des seltenen Crisponi Syndroms aufklären. Hierzu werden neuronale Zellkulturen von einer Crlf Knockoutmaus und von Wildtyp Tieren etablieren. Anschließend werden wir die Expression und Synthese noradrenerger Proteine sowie die Proteolyse in diesen Zellen überprüfen. Der Pathomechanismus des Crisponi Syndroms kann nur an Zellen mit demselben Gendefkt wie bei den Crisponi Patienten untersucht werden. Daher sind wir gezwungen mit neuronalen Zellen aus den genetisch veränderten Mäusen zu arbeiten.
Biometrisch wurde die Mindestanzahl an Mäusen berechnet, die notwendig ist, um aussagekräftige Ergebnisse über die Wirksamkeit der Therapien zu erhalten.
Es ist bekannt, dass die verwendeten Mäuse direkt nach der Geburt aufgrund von Schluckstörungen versterben. Unsere Tiere werden nicht geboren, da das Versuchsende auf den Embryonaltag E 17 festgelegt ist. Die Generierung der Tiere erfolgt nur zu Untersuchungszwecken über eine heterozygote Zucht.

6.2 Doc 6184

Cannabinoidrezeptor 2 - Expression bei **inflammatorischen** Prozessen der Maus
In diesem Versuchsvorhaben möchten wir den CB2-Expressionsverlauf mittels unserer Reportermauslinie, CB2-GFPTg, während inflammatoricher Prozesse im Gehirn und pe-

ripher untersuchen. Dabei sind für uns die EAE als **Multiple Sklerose**-Modell und die LPS-Applikation als **Sepsis**-Modell besonders interessant, da hier der CB2-Rezeptor eine wichtige Rolle spielt. Das in unserem Labor generierte Mausmodell, CB2-GFPTg, stellt ein gutes Werkzeug zur Analyse der CB2-spezifischen Proteinexpression im generellen und entzündlichen Verlauf von Erkrankungen dar. Die Ergebnisse können ein völlig neues Licht auf die Funktion von CB2 während der Pathogenese der **Sepsis und der Multiplen Sklerose** werfen und zukünftig neue Behandlungsmethoden ermöglichen.
Intraperitoneale Injektionen führen zu einer punktförmigen Verletzung, die nur wenig schmerzhaft ist.Die einmalige Applikation von LPS ist als mäßige bis erhebliche Belastung anzusehen. Die zu erwartenden Belastungen bei der intrakranialen Injektion sind von der Dauer und Intensität her als mittelgradig einzustufen. Die Immunisierung mit Freund?s Adjuvans stellt eine mäßige Belastung dar, da sie subkutan verabreicht, zu leichten und lokal begrenzten Entzündungsreaktionen führen kann. Die Belastungen für die Tiere bei der EAE überschreiten die Dauer von 14 Tagen nicht, wodurch sie als mittelfristig einzustufen sind. Tötung erfolgt durch zervikale Dislokation und Perfusion.
Der Einfluss von CB2 auf den Krankheitsverlauf der **Multiplen Sklerose** oder während der Sepsis kann nur mit einem der Krankheitssituation möglichst ähnlichem Modell erreicht werden. Andere Versuchsansätze, wie z.B. Zellkulturexperimente, sind nicht anwendbar, da diese die Komplexität und Interaktion unterschiedlicher entzündungsbedingter Vorgänge im ZNS und auch peripher nicht ausreichend widerspiegeln können. Die komplexen Interaktionen von diversen Zelltypen können in vitro nicht nachvollzogen werden.
Die geschätzte Anzahl der Tiere basiert auf unseren derzeitigen Erfahrungen auf dem Gebiet der Verhaltensanalysen.Wir haben eine ausführliche statistische Analyse durchgeführt, um sicherzustellen, dass die minimale Anzahl von Tieren verwendet wird, um signifikante Ergebnisse zu erzielen. Nach Möglichkeit werden die in vitro Studien bevorzugt eingesetzt.
Die in diesem Projekt verwendeten Mäuse werden in einer modernen Tierhaltung unter SPF-Bedingungen gehalten. Die Tiere werden in Gruppe gehalten und mit Einstreu versorgt. Die Behandlungen der Mäuse mit LPS sowie die EAE-Versuche führen zu einer mäßigen bzw. geringen Belastung der Tiere. Die erforderlichen Dosierungen und Versuchsabläufe werden so gewählt, dass diese Belastungen unter Beachtung des Untersuchungszieles so gering wie möglich sind.Zur Schmerzlinderung nach der intrakranialen Injektion wird den Mäusen alle 6 Stunden Buprenorphin subcutan (0,1 mg/kg) verabreicht.Sollten die Tiere eine Infektion erleiden oder ernsthaft erkranken, werden sie vorzeitig schmerzfrei getötet.

6.3 Doc 6158

Einfluss von Adipositas auf eine mitochondriale Dysfunktion
Mitochondrien haben in der eukaryotischen Zelle zahlreiche essentielle Funktionen und

generieren nahezu den gesamten Anteil an intrazellulärer Energie in Form von ATP. Sie bilden in der Zelle ein dynamisches tubuläres Netzwerk, welches durch ständige Fusions- und Teilungsprozesse aufrecht gehalten wird. Der Mechanismus der mitochondrialen Dynamik ist noch nicht vollständig aufgeklärt. Allerdings wurde bereits deutlich, dass mit einer morphologischen Veränderung einhergehende mitochondriale Funktionsstörungen beim Menschen mit einer Vielzahl von Krankheiten assoziiert sind. Aktuelle Studien geben Hinweise darauf, dass solche Funktionsstörungen, in den Beta-Zellen der Langerhansschen Inseln des Pankreas die Entstehung des **Diabetes mellitus Typ 2** begünstigen. Neuste Studien zeigen, dass eine chronische Exposition mit gesättigten freien Fettsäuren, wie sie bei einer Adipositas vorliegen, zu einer Beta-Zell-Dysfunktion mit einhergehender mitochondrialen Dysfunktion führt. Eine vergleichbare Dysregulation der mitochondrialen Dynamik wird als Ursache der Entstehung einer Insulinresistenz diskutiert. Es ist das Ziel dieses Projektes, durch eine kalorienreduzierte Diät ehemals adipöser C57BL/6ob/ob Mäuse die mitochondriale Dysfunktion und einen möglichen Zusammenhang bei einer Beta-Zell Dysfunktion und Insulinresistenz zu untersuchen. Welchen Einfluss die Adipositas auf mitochondriale Funktionsstörungen hat und ob die morphologischen Veränderungen reversibel sind, ist bislang noch unbekannt. Dieser Aspekt soll im beantragten Versuchsvorhaben an C57BL/6ob/ob Mäusen untersucht werden.
Die Belastung der Tiere während der retrobulbären Blutabnahme (in Narkose) wird als mäßig eingeschätzt. Die i.p.-Injektionen (ohne Narkose) werden als geringfügig eingestuft. Da die Mäuse im Rahmen der Blutzuckermessung nicht merklich auf die Penetration der Schwanzspitze reagieren, wird davon ausgegangen, dass dies ebenfalls nur eine geringfügige Belastung darstellt.
1.Im Rahmen des Projektes soll untersucht werden, ob eine Gewichtsabnahme beim Adipositas-Mausmodell eine mitochondriale Dysfunktion verbessern und somit einer Insulinresistenz entgegenwirken kann. Welchen Einfluss Adipositas auf mitochondriale Funktionsstörungen hat ist bislang noch unbekannt. Dieser Aspekt soll imbeantragten Versuchsvorhaben an C57BL/6ob/ob Mäusen untersucht werden.2.Die geplanten in und ex vivo Untersuchungen sind wegen der komplexen Interaktionen durch in vitro Methoden nicht ersetzbar.
Die nach Versuchsende entnommenen Organe werden für verschiedene Fragestellungen verwendet. Dadurch wird die Zahl der verwendeten Tiere auf das unerlässliche Maß beschränkt.
1.Das Leid der Versuchstiere wird durch eine gute Tierhaltung, die sachgerechte Aus- und Weiterbildung der beteiligten Personen und gewissenhafte Versuchsplanung so gering wie möglich gehalten.2.B6.V-lepob Mäuse leiden an einem absoluten Leptinmangel, welcher zum Fehlen des Sättigungssignals und unkontrollierter Nahrungsaufnahme führt. Die Mäuse entwickeln Adipositas, welches von Hyperglykämie, Hyperinsulinämie und Insulinresistenz begleitet wird.

6.4 Doc 6183

Test von biologischen Matrices zur Rekonstruktion der Augenoberfläche
Vernarbende Erkrankungen der Augenoberfläche entstehen zum Beispiel nach Verletzungen, Tumorerkrankungen oder bei Autoimmunerkrankungen. Hierbei kommt es zu Fremdkörpergefühl bis hin zu starken Augenschmerzen und zur Sehminderung bis zur Erblindung. Bei schwerer Ausprägung ist zur Therapie die Wiederherstellung einer intakten Augenoberfläche, also der **Hornhaut** und der Bindehaut, erforderlich. Hierfür stehen derzeit keine Transplantat-Materialien zur Verfügung, welche die Voraussetzungen der mechanischen Stabilität, Biokompatibilität, sowie reproduzierbaren Herstellbarkeit erfüllen.In diesem Versuchsvorhaben sollen vier verschiedene Gewebe für die Rekonstruktion der Hornhaut und Bindehaut in vivo getestet werden. Bei erfolgreicher in vivo Testung können diese Gewebe für Pilotversuche im Menschen eingesetzt und potentiell als reproduzierbar herstellbare Ersatzgewebe zur Rekonstruktion der Augenoberfläche genutzt werden.
Je Tier wird ein Auge kontrolliert chirugisch verletzt. Es entsteht hierbei ein Schaden der Hornhaut oder der Bindehaut. Die Verletzungen werden direkt chirurgisch versorgt und die Augen mit schmerzlindernden und antibakteriellen Augentropfen behandelt. Dadurch werden Schmerzen reduziert und das Risiko für eventuelle Infektionen verringert. Die Orientierungsfähigkeit der Tiere wird nicht beeinträchtigt, da immer ein Auge unbehandelt bleibt. Da es sich um Finalversuche handelt (die Tiere werden euthanasiert), entsteht für die Tierekein anhaltender Schaden. Während der Versuche ist eine mittelgradige Belastung zu erwarten (gemäß Ahang VIII, Richtlinie 2010/63/EU). Eine schwere Belastung stellt ein Abbruchkriterium dar.
Den geplanten Untersuchungen sind erfolgreiche in vitro Arbeiten vorangegangen. Die Testung der Handhabbarkeit, der chirurgischen Anwendbarkeit, der Einheilung in das Gewebe und der Einfluss von externen Faktoren wie Lidschlag und Augenbewegungen sind nur im vitalen Organismus, also im Tiermodell, testbar. Daher kann nur im Tierversuch getestet werden, ob die Gewebe für eine Anwendung im Menschen geeignet sind.
Es erfolgte eine biometrische Planung mit Hilfe des Programms GPower 3.1.Hierdurch wurde die Anzahl der Tiere in einer Gruppe auf das notwendige Maß reduziert, um eine biologisch relevante Differenz zu erreichen. Die Kontrolltiere wurden auf die Mindestzahl begrenzt.
Das Kaninchen ist ein etabliertes Modell für die Untersuchung von Augenerkrankungen. Das Kaninchenauge hat außerdem eine angemessene Größe für chirurgische Eingriffe und Augenuntersuchungen sowie die Tropftherapie nach der Operation, so dass diese in optimaler Weise erfolgen können. Ein kleineres oder geringer entwickeltes Tier eignet sich nicht für die geplanten Versuche. Es wird immer nur jeweils ein Auge des Tieres operiert, so dass die Orientierungsfähigkeit erhalten bleibt. Der chirurgische Eingriff bleibt lokal auf die Hornhaut oder Bindehaut begrenzt und der Lidschluss wird nicht beeinträchtigt, dadurch werden Schmerzen reduziert. Durch eine angepasste lokale Augentropftherapie

werden Infektionen, Entzündungen und Immunreaktionen verhindert.

DECLARATION

I, Manjil Shrestha, hereby declare that the work presented herein is my own work completed without the use of any aids other than those listed. Any material from other sources or works done by others has been given due acknowledgement and listed in the reference section. Sentences or parts of sentences quoted literally are marked as quotations; identification of other references with regard to the statement and scope of the work is quoted. The work presented herein has not been published or submitted elsewhere for assessment in the same or a similar form. I will retain a copy of this assignment until after the Board of Examiners has published the results, which I will make available on request.

03.09.2020

Manjil Shrestha | Date