Universiteit Leiden

Master Computer Science

Classifying general practitioner notes using support vector machines and pre-trained language models

Name: Marcus Abukari
Student ID: s1433911
Date: 13-09-2020

1st supervisor: Suzan Verberne
2nd supervisor: Matthijs van Leeuwen
3rd supervisor: Hine van Os

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1, 2333 CA Leiden
The Netherlands

Abstract

General practitioners (GPs) use the International Classification of Primary Care (ICPC) system as a tool for classifying the reason for encounter (RFE) and medical diagnosis for episodes of care. These codes are recorded after every patient visit, accompanied by a short summary in free-text form. Clinical experts within the LUMC observed under-reporting and general noisiness of these codes in the electronic records, while the corresponding free-text entry often contains valuable information such as the diagnosis or elements strongly related to a diagnosis such as prescriptions. Both the annotation of missing data and automatic labeling of future records requires an automated classification tool, which is the focus of this research project. For developing this classification model the relatively new natural language processing technique BERT will be compared with a more traditional approach using the Support Vector Machines (SVM). The results show both models reach a 90% precision, while the SVM outperforms the BERT model on the recall and F1-score. While the results based on these commonly used metrics are similar, the BERT model faces several challenges compared to the SVM most notably the failure to predict under-supported classes while also requiring substantially more computing resources and development time. Therefore the SVM is the preferable technique given the data and problem as presented in this project.

Contents

Chapter 1

Introduction

1.1 Problem statement

In the Netherlands, general practitioners (GPs) report patient visits in an electronic medical record. One of the attributes of this record is the diagnosis made by the GP summarizing the nature of the visit which is encoded according to the International Classification of Primary Care (ICPC). This standardized coding system is able to cover a wide spectrum of possible diagnoses. Additional information which does not fit in this encoding system, or any other additional information the GP finds relevant, can be added manually in a free-text field and is stored alongside the ICPC code in the same record.

ICPC codes are not human interpretable and GPs either have to memorize the codes or use external tooling in order to find the suitable ICPC code. Clinical experts within the LUMC point to external factors such as time pressure and manual errors as the cause for the under-reported and noisy ICPC codes. This hypothesis is supported by the observation of certain high interest ICPC codes (e.g. hypertension and migraine) being under-reported when their prevalence in these medical records is compared with the prevalence in other national studies. In addition to under-reporting, there is also no mechanism in place to verify the correctness of registered ICPC codes. Both the noisiness and the quality of the reported ICPC codes result in a methodological limitation in the use of these ICPC codes for other observational studies. Furthermore, given the volume, variety and velocity of these electronic records, it is not feasible to manually encode missing ICPC codes or verify existing instances. Therefore, this problem requires an automated natural language processing solution using machine learning, capable of accurately determining the diagnosis based on the free-text notes.

There are several challenges in classifying text in general, but also for clinical or biomedical text data [5]. Given the active developments in the machine learning

domain there are different approaches to which yield different outcomes depending on the problem. In general, prior academic research on the topic report that it is possible to classify clinical text data showing that machine learning models are able to learn from the data with varying levels of success given different approaches and models [17] [13] [14]. Apart from the classification accuracy other factors also play a role in the search for the most optimal approach. While novel techniques, such as deep learning, can show incremental accuracy improvements they can face other challenges such as increased training time or their improvements are tied to the size of the dataset [13].

In recent years there have been several advancements in the natural language processing (NLP) field. Improvements on language models are continuing to push state-of-the-art benchmarks results and the concept of pre-training and transfer learning allow for models to be reused and distributed which increases the accessibility of these techniques [3]. The Bidirectional Encoder Representations from Transformers (BERT) language model is one the the most recent developments to the natural language processing field. However, benchmark results do not reflect the performance on real life tasks. Some argue for BERT to be used as the default technique for NLP tasks while other research show that, especially for classification tasks, the results of the BERT model are comparable to traditional machine learning techniques [6] [12].

The machine learning and NLP domain enjoy active development and academic research. And while these techniques are also applied in the medical field on clinical text, there is little to no consensus on the (overall) best performing methods. In addition to this, there is also little research done on applying these techniques on Dutch clinical text data. This research project will attempt to contribute to the search for the most optimal approach on Dutch data while also providing the LUMC with the foundation on applying these techniques in the future to solve the problem of missing ICPC codes in their medical records.

1.2 Goal of this research

The main goal of this research is to develop a model using machine learning capable of predicting ICPC codes given a free-text entry from a patient visit to the GP. In practice, the model and the corresponding development code should provide a scalable solution for annotating missing data and predicting the ICPC code for new entries while also providing the groundwork for future applications such as anomaly detection.

In order for this research project to also provide academic relevance in the field of

natural language processing, the performance of these (relatively) new techniques will be compared to more traditional machine learning algorithm: the Support Vector Machine (SVM). The reason for choosing the SVM specifically in this research project is due to its good general performance for text classifications tasks [17] [8]. And also due to its performance on multi-label classification problems compared to other algorithms[16]. The research question is therefore formulated as follows:

"To what degree can machine learning algorithms predict ICPC encoded diagnoses based on free-text notes and how do traditional techniques compare to pre-trained language models?"

1.3 Thesis overview

The research project is divided in several chapters. Chapter 2 will elaborate on the used techniques and algorithms, such as the nature of the ICPC codes as well as the machine learning techniques. The chapter will conclude with an analysis on existing literature relevant for this project. In Chapter 3 an in-depth exploratory data analysis will be conducted and the corresponding preprocessing steps will be discussed. The chapter will also describe the methodology for both models and the evaluation metrics used to evaluate and compare their performances of the models on the data. After the methodology the results for both models will be reported separately followed by the comparison discussion. The final part of the report will evaluate the practically and applicability of the best performing model in the medical domain. The report will conclude with a discussion about the results followed by the conclusion.

Chapter 2

Background & related work

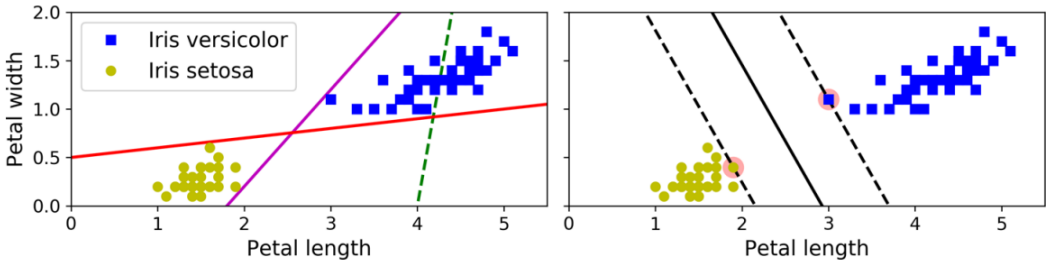
2.1 ICPC codes

Policy-makers and other providers in healthcare require accurate information on the epidemiology of their communities. Without this information it is difficult to get a high level overview of the epidemiological environment which hinders the ability to improve health care services. In order for health providers to effectively record information as part of their clinical routine, as well as standardizing communication of this information to other areas of the health domain, a robust classification tool for diagnoses needed to be developed.

In 1987, the International Classification of Primary Care (ICPC) was developed for this purpose as a tool to order the domain of general practitioners. It was designed from family medicine data in order to not only classify but also define relationships between the nature of episodes of care. An episode of care refers to a health problem from the first presentation to the health care up until the last presentation of the same problem. After growing in popularity, the World Health Organization (WHO) accepted the format as a member of the Family of International Classifications (WHO/FIC). The ICPC allows health care providers to classify three important elements of the health care encounter: reasons for encounter (RFE), diagnoses or problems, and the further process of care.

The ICPC is based on a bi-axial structure in which symptoms, procedures and diseases are coded into chapters represented by letters and recorded in the first axis [2]. On the second axis, a two digit numeric code refers to one of seven diagnostic components. For example, Component 1 provides rubrics for symptoms and complaints while Component 7 reflects the diagnosis or disease for each chapter. This component will only be used when there is sufficient information to support this medical diagnosis in order to record it [22].

Figure 2.1: 4 decision boundaries of different classification models trained on the iris dataset. The plot on the right shows the decision boundary produced by the SVM.



2.2 Models for text classification

2.2.1 Support vector machines

A support vector machine (SVM, or support-vector-network) is an algorithm for supervised machine learning capable of performing linear or nonlinear classification, regression and outlier detection and has shown to be a robust algorithm for learning text classifiers [8]. As further described in Chapter 3, in this research project the algorithm will be trained to solve a multi-label classification problem. To illustrate the workings of the SVM, Figure 2.1 shows the plot of the famous iris dataset with two classes which are linearly separable [4]. The left plot shows three different linear classifiers with their respective decision boundaries. The classifier represented by the dashed line fails to separate the two classes from each other while the other two classifiers (the red and purple lines) perform this task with 100% accuracy. The right plot shows the decision boundary of the SVM. Compared to the classifiers of the left plot, the decision boundary of the SVM stays as far away from the closest training instances of the two classes. This characteristic of the SVM leads to a model that generalizes well and will therefore be more likely to perform on unseen data. These closest training instances, highlighted in the right plot, are also referred to as the *support vectors* hence the name Support Vector Machine.

In the figure the two classes are linearly separable, however in reality this is often not the case due to noise, outliers or just due to the nature of the data. In this case there needs to be a balance between maximizing the distance between the classes and the number of violations over the decision boundary. This value can be adjusted to find the best model for a specific problem while it also provides the possibility the reduce overfitting.

2.2.2 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) attempt to solve some of the limitations experienced with neural network types such as Recurrent Neural Networks (RNN) and Long Short-Term Memory Networks (LSTM) which were often used for NLP tasks. Given a dataset of sentences for a NLP task, RNNs and LSTMs iterate over each word in the sentence in one direction (in this case left to right) and derive the meaning of the words from all preceding words in the sentence. The networks typically consist of an encoder and a decoder where the encoder converts the sequence to a vector representation while the decoder converts the same vector to a sequence of words. RNNs and LSTMs require two inputs: an input vector (a vectorized representation of a word) and a hidden state. At each training step in the neural network both inputs are processed and two new output vectors are produced: the output vector and a new hidden state. This new hidden state is then used for processing the next input vector and the cycle continues. This process leads to the first limitation of these networks as they are unidirectional and can therefore only defer the meaning for a word from the preceding tokens in the sequence. The second limitation stems from regenerating the hidden state at each time step in the network. For a given input vector this means the preceding tokens are summarized by the hidden state and the final output of the encoder is the hidden state from the last step. This leads to a situation where it becomes difficult to capture the relations between tokens that are separated by a greater distance in a sentence.

Luong et al. argued that the use of this encoder-decoder architecture bottlenecks the improvement of language model performance and proposed a new technique for interpreting longer sentences which is referred to as attention [11]. The attention mechanism is used as an improvement upon the previously describer encoder-decoder architecture. Using a sequence-to-sequence task as an example, the two differences between a classic sequence-to-sequence model and the attention model can be summarized as follows:

- Instead of passing only the last hidden state from the encoder to the decoder, the attention model passes *all* hidden states from the words in the input sentence to the decoder.

- The decoder uses a scoring function to assign scores to all hidden states it receives from the encoder. These scores are softmaxed in order to amplify high scores and penalize low scores. This is were the *attention* terms originated from since these hidden states represent input tokens and this mechanic puts extra emphasis (attention) on these tokens. Lastly, the hidden states are multiplied by their softmax score and summed up creating

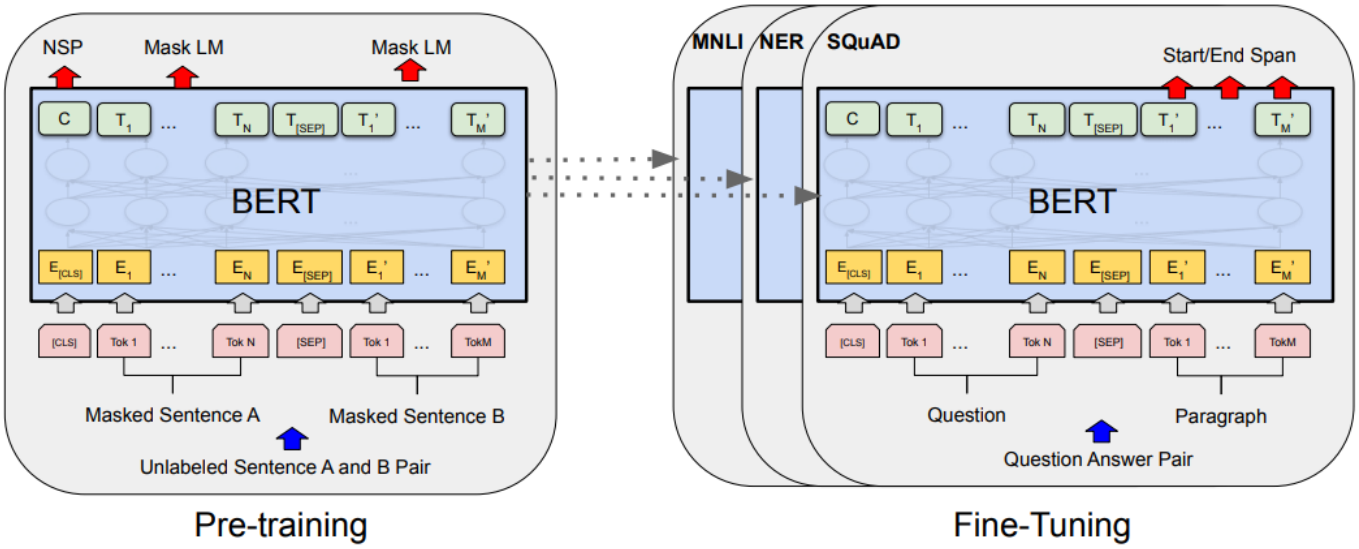


Figure 2.2: The image on the left shows the unsupervised pre-training task for generating the word embeddings, the image on the right shows different fine-tuning tasks for example question answer pairs.

the context vector for the remainder of the decoding task.

In 2017 Google published the paper "Attention is all you need" where they introduced a new network architecture based on the attention mechanic called the Transformer [21]. This network architecture does not rely on recurrence or convolutions but uses a slightly modified attention mechanic called self-attention. The main benefit of this approach is that the training tasks are parallelizable which requires significantly less processing power and time to train, while also outperforming the best models from literature [21].

The previously described techniques and mechanisms have lead to the introduction of a new language representation model: the Bidirectional Encoder Representations from Transformers (BERT) [3]. The implementation of BERT models for NLP tasks consists of two steps: *pre-training* and *fine-tuning*. In the pre-training step the model is trained on a large corpus of unlabeled data for a given language, for example the full English Wikipedia, wordpiece vocabularies, or other custom sources. Section 3 will elaborate further on the chosen pre-trained models for this research project and their origin. For the fine-tuning, the pre-trained language models parameters are fine-tuned using labeled data for a specific downstream task such as classification or named entity recognition (NER). Downstream tasks is what the field calls those supervised-learning tasks that utilize a pre-trained model or component. Each fine-tuning task results in a task-specific model, while the pre-trained models can be reused for other fine-tuning tasks given that the

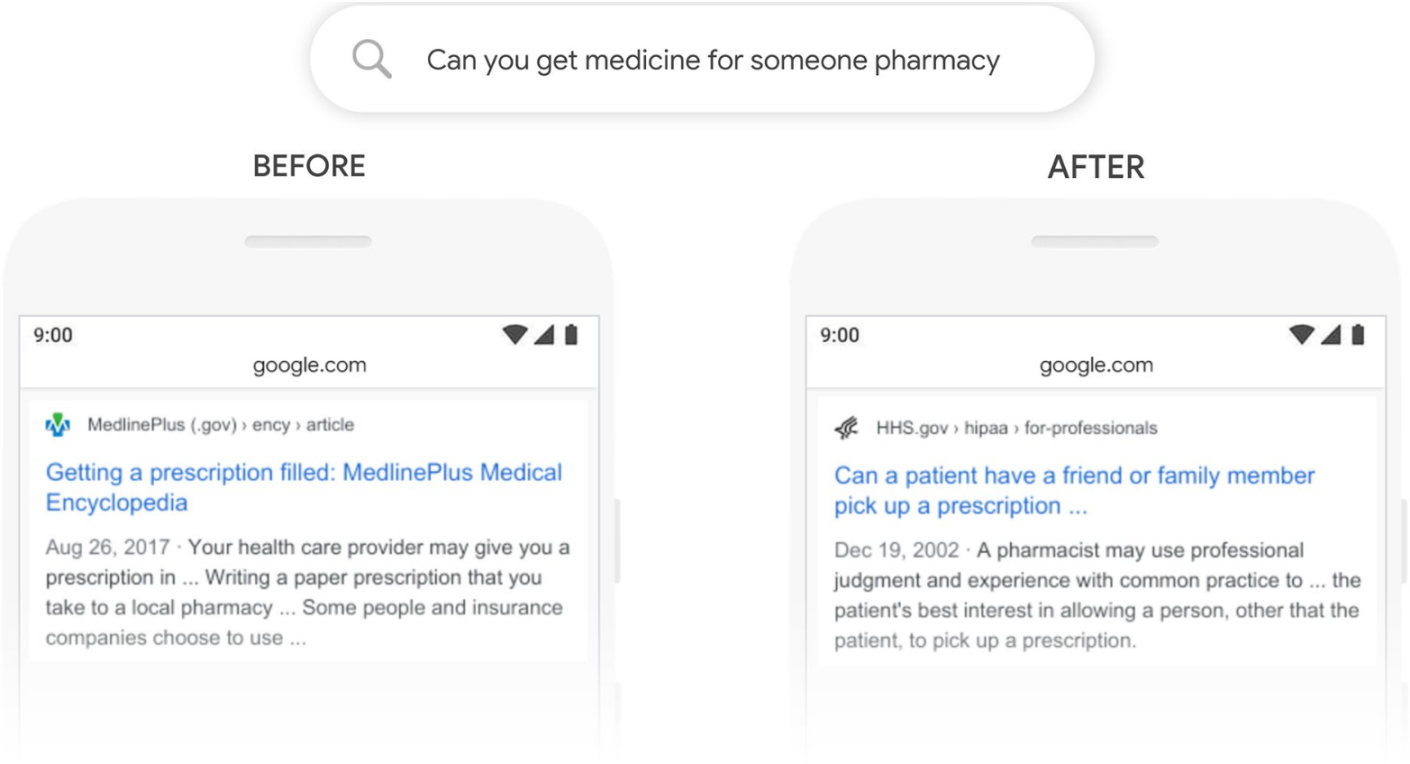


Figure 2.3: The before example (left) fails to capture the meaning of "for someone" while the example on the right using BERT does and provides relevant search results [7]

language of the pre-trained model aligns with the downstream task. BERT can be used for a wide variety of NLP tasks such as classification, question and answering and named entity recognition. Figure 2.2 visualized the pre-training and fine-tuning task for a question and answer pair task.

In October 2019 Google published a blog explaining the improvements BERT made within the natural language field for their Search functionality [7]. Users of the Search product are not always confident in how they formulate their question. This is partly due to the fact that most NLP implementations struggle with more natural conversational queries. For this reason, Google noticed many users falling back to building queries using keywords instead of conversational language. In most cases, this approach yields better search results even though it is not how people naturally ask questions.

Within conversational queries and general complex natural language prepositions such as "for" or "to" carry significant meaning. Google used the following example in their introduction blog. Consider the search query "2019 brazil traveler to usa

need a visa". In this context, the preposition "to" refers to traveling from Brazil *to* the USA, and not the other way around. Previously, Google's algorithms and most other NLP techniques struggled to grasp the meaning of "to" since it depends on the words used and the order of these words in the sentence. Figure 2.3 shows a second example. The before result shows that the "for someone" is not properly understood by the original model while on the right the BERT model is able to detect this subtle nuance and provide the user with the correct results.

2.3 Prior research

In the paper "Classifying Encounter Notes in the Primary Care Patient Record", Rost et al. conducted research with a problem similar to the problem statement in this project as they attempt to classify free-text of clinical encounters with an ICPC code [17]. The data used consisted of roughly 500,000 unique encounters from a medium-sized general practice office in Norway however only 175,167 encounters were fit for training. The resulting SVM classifier was able to correctly predict 994 of the 2,000 instances in the test set yielding an overall accuracy rate of 49.7%. In comparison, a classifier which assigns the most frequent ICPC code in the training dataset to each test instance, has an accuracy of 20.8%. The authors argue that improvements can be made with a more balanced dataset, but also with improved text processing techniques such as stemming or lemmatization as the free-text not only varies in length but is also subject to spelling errors and (medical) abbreviations. The noisy characteristic of the free-text is also visible in the data used for this research project and therefore these text processing techniques could benefit the models performance. The most important difference compared to this research project is that Rost et al. use all 726 distinct ICPC codes divided in a multi-class classification problem with two classes whereas this research project focuses on a small subset of target ICPC codes. Therefore while the problem is similar, the results will be difficult to compare.

Pre-trained language models are often trained on large corpora such as Wikipedia or a collection of literature [20]. This results in models which can be trained for a variety of downstream tasks since their embeddings are not domain specific. In the context of this research, a language model trained on Wikipedia should include a wide spectrum of clinical text however the fraction of relevant (clinical) text is small which may increase the time of fine-tuning and decrease the performance due to time constraints.

Lee et al. evaluated the performance differences between general BERT models and BERT models trained on biomedical data (BioBERT) of which the word em-

beddings are derived from domain specific (clinical) sources [10]. The BioBERT model was trained on a collection of PubMed abstracts and PMC full-text articles and fine-tuned for three popular biomedical text mining tasks: Named Entity Recognition, Relation Extraction, and Question Answering. The results show that the BioBERT model outperforms the standard BERT model on all biomedical text mining tasks even though the improvement are marginal. In the paper "Publicly Available Clinical BERT Embeddings" Alsentzer et al. perform similar research using the MIMIC-III critical care database [9] [1]. The MIMIC-III database contains information on patients admitted in to critical care units at a large tertiary hospital. The data includes numeric attributes such as fluid balance, procedure codes and diagnostic codes but also natural text data originating from nursery and physicians. In the paper they argue the relevance of using this database compared to the sources used for the BioBERT model as clinical narratives (e.g. physician notes) have known differences in linguistic characteristics compared to the general text from the default BERT model but also from the non-clinical sources used for the BioBERT model. In the paper, the BERT model is trained on all note types in the model "Clinical BERT" while a variety is trained on the discharge summary of patients and is referred to as the "Discharge Summary BERT".

Table 2.1: A comparison of the performance of BERT models trained on domain specific (biomedical) text and the default BERT model [1]

Model	MedNLI
BERT	77.6%
BioBERT	80.8%
Clinical BERT	80.8%
Discharge Summary BERT	80.6%
Bio+Clinical BERT	82.7%
Bio+Discharge Summary BERT	82.7%

Table 2.1 displays the results of the experiments using different combinations of the Clinical BERT, Discharge Summary BERT and BioBERT model for the MedNLI natural language gauge inference task. The results show that all of the models trained on a clinical or biomedical source outperform the default BERT model. The paper also reported significant computational costs as they estimate the embedding modeling procedure to have taken approximately 17 - 18 days. Even though the paper and resulting model are publicly released, these pre-trained models can not be reused in this research project as they are trained on English text.

Chapter 3

Data & Methods

In this Chapter the data used will be analyzed and the methods used for training both models will be further explained. Sections 3.1 and 3.2 will provide and overview and a in-depth analysis of the data respectively. Section 3.3 will describe the preprocessing steps taken the convert the unprocessed dataset to a dataset fit for both algorithms to train on. The proceeding Section 3.4 will describe the training for both models. This section is divided in two subsections for both models as there is some model specific tasks. In the last part, Section 3.5 the metrics used to compare the results of both models will be explained.

3.1 Data source

In order to build a model and answer the research question, the LUMC provided a large dataset containing visits to general practitioners. The dataset is a product of the previously conducted NEO study which started in 2008. The NEO study is a population-based, prospective cohort study designed to investigate pathways that lead to obesity-related diseases. Men and women living in the greater area of Leiden were invited to respond only if their age is between 45 and 65 years and if they had a self-reported BMI of $27kg/m^2$ or higher. In addition, all inhabitants aged between 45 and 65 years from one municipality (Leiderdorp) were invited to participate, regardless of their BMI (n=8,229 invited, response rate 20.3%). In total, 6.671 participants agreed to join the study.

This research project is conducted on a static copy of this dataset which was exported to CSV format. Before exploring the data in depth in Section 3.2, Table 3.1 shows an overview of all columns in the CSV dataset with their corresponding data type and a short description.

The text column contains a short description summarizing the patients visit. This

Table 3.1: An overview of the column names, data type and an optional description of the unprocessed dataset

Column name	Type	Description
ID	int	Instance identifier
NEONR	int	Patient identifier
soep	string	-
volgnummer	float	-
icpc	string	ICPC code
probleemnummer	float	-
actiecode	string	-
consultsoort	string	-
icpcprobleem	string	-
contactdatum	string	Date in `%Y%m%d` format
tekst	string	General practitioner notes
artscode	string	-
actietxt	string	-
mdwcode	string	-
consulttxt	string	-

text is entered in the system by the GP manually. In general, there are no input requirements regarding null values, input length, or formatting rules and in practise, the use of the field varies between GPs. Early observations show that the text data is relatively noisy. GPs often enter a small amount of keywords regarding the nature of the visit accompanied with the description of a prescription and dosis.

3.2 Exploratory data analysis

This section contains an in-depth exploratory data analysis on the raw data. The goal of this analysis is to provide a high level overview and visual summary of the data. Furthermore, it will also inspect the data and highlight areas that may hinder the performance of machine learning models later in the project. This section will limit itself to exploring the data while Section 3.3 describes the data cleaning and transformation.

The first step of the analysis is to determine the subset of the data which can be used during this research project. As described in Chapter 1, the goal of this research is to predict the ICPC code, given the text notes of the general practitioner. The most important columns are the ICPC code and the text, and as will be de-

Table 3.2: An overview of missing values and duplicate instances in the data

Field	Absolute count	Percentage of total
Missing ICPC values	3,043,013	89.48%
Missing text values	36,103	1.06%
Missing NEONR values	0	0.00%
Missing dates values	618	0.02%
Duplicate instances	78,056	2.30%
Total instances	3,400,946	100.00%
Remaining instances	341,464	10.04%

scribed in Section **??**, the project will also use the "NEONR" (patient identifier) and "contactdatum" columns for data transformation purposes. Table 3.2 shows a summary of the missing values and duplicate instances. An instance is marked duplicate when the combination of all four described target fields (ICPC, tekst, NEONR and contactdatum) have multiple occurrences in the dataset. Note that the values of the missing fields and duplicate instances (the absolute count and the percentage of total) are not accumulated but calculated independently from the rest of the table. The table shows that roughly 10% of the dataset, equal to 341,464 instances, can be used for training the ICPC classifiers.

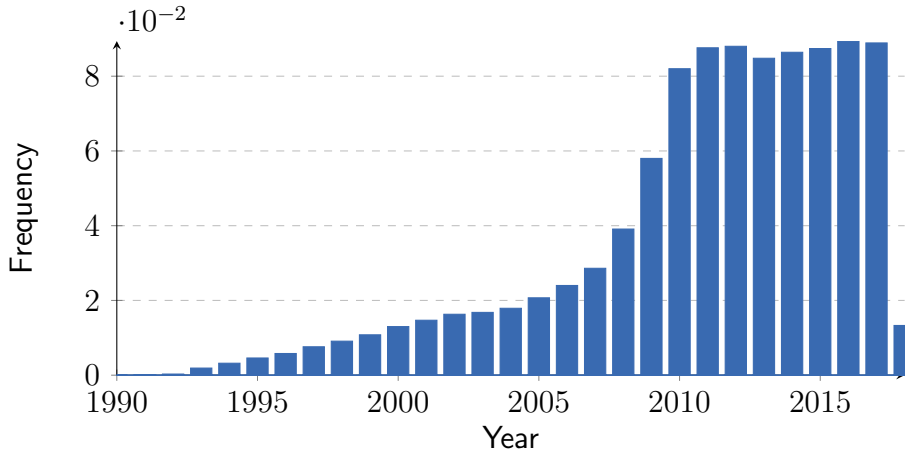
Table 3.3: Distinct ICPC codes and NEO numbers

	Unique	Total
ICPC codes	1,784	341,464
ICPC codes (first 3 characters)	875	341,464
NEO numbers	4,807	341,464

As described in Chapter 2, the ICPC codes follow a predefined format. The main component consists of the first 3 characters of the code which captures the medical diagnosis type for the specific GP visit. Optionally, an extra number can be added in addition to the main component separated by a dot. This number indicates a variation on the main diagnosis. For example, the ICPC codes T90.01 and T90.02 refer to diabetes mellitus type 1 and diabetes mellitus type 2 respectively. In the dataset, all three variations (T90, T90.01, T90.02) are used which increases the distinct number of ICPC codes. Table 3.3 contains an overview of the distinct values of both the ICPC values variations and the distinct number of NEO numbers. The table shows that roughly half of all unique ICPC codes are variations on the main diagnosis. Furthermore, since the NEO numbers serve as

unique patient identifiers, this value represents the number of distinct patients in the dataset after removing duplicate and missing data.

Figure 3.1: Relative frequency histogram for the appointments per year



The cleaning process involved removing instances containing missing values and removing duplicate rows. Table 3.2 shows that 90% of the dataset contains either missing values or are a duplicates and these instances can not be used for training the classifier. Figure 3.1 shows the distribution of remaining instances in the dataset by year. The figures shows that the majority of the visits are relatively new as they took place after 2010. There is also a smaller subset of data ranging back to 1990. In order to investigate the use of the older data, the next part will analyze the follow up time of patients in the dataset.

Figure 3.2: Relative frequency diagram for the follow up time in months

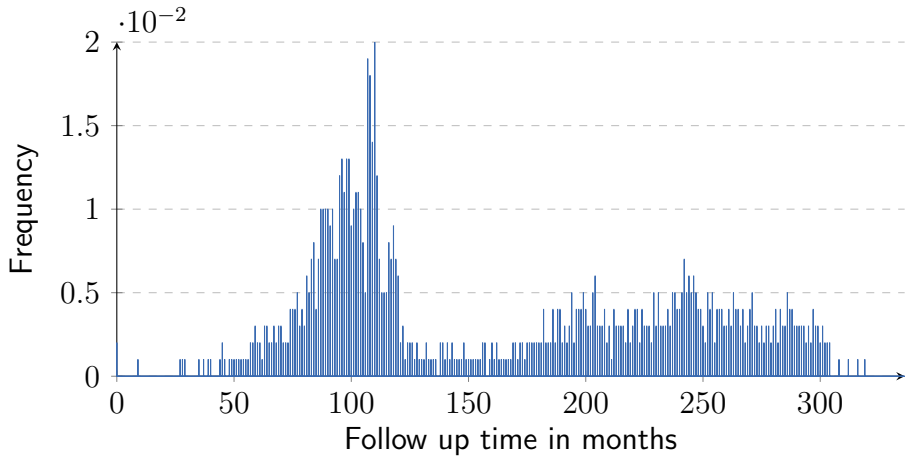
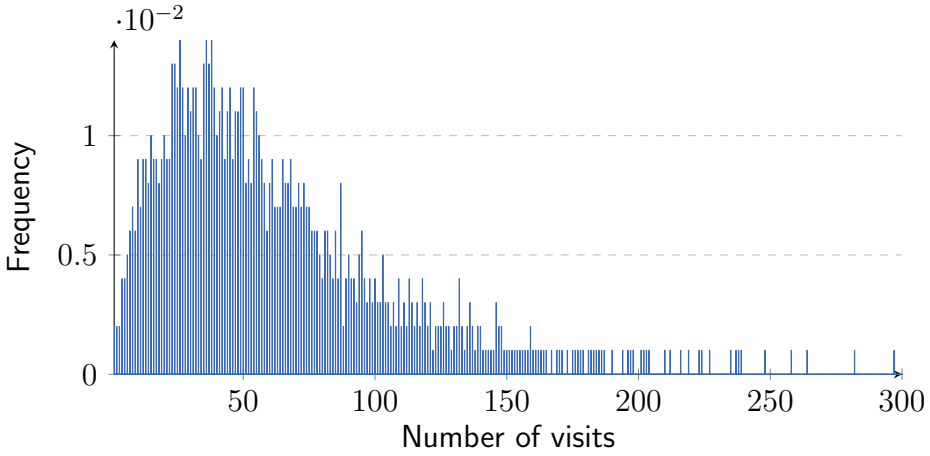
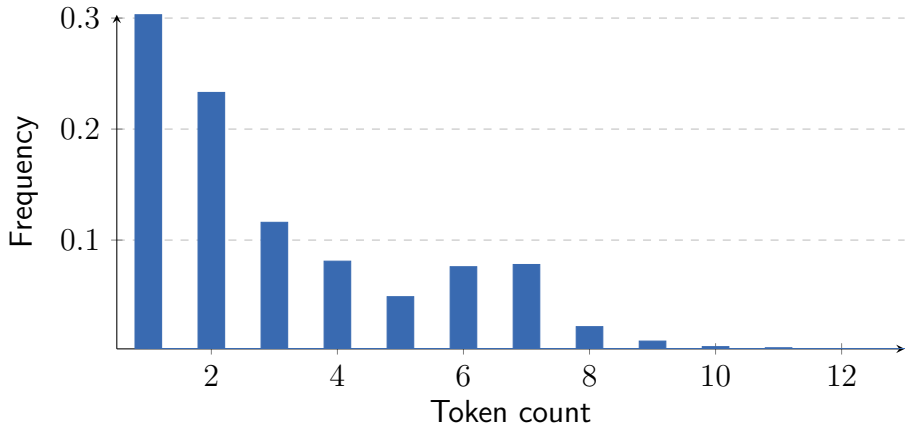


Figure 3.3: Relative frequency histogram for the total visits per patient



The follow up time (in the context of this part of the exploratory data analysis) is the duration between the first and last registered visit in the dataset. This metric reflects the time period patients are active within the data. However, a long follow up time does not show the density of visits within this time frame, therefore we are also interested in the visits per patient. Figure 3.2 shows the follow up time distribution in months while Figure 3.3 contains the distribution of the number of visits per patient.

Figure 3.4: Relative frequency histogram for the token count per appointment



The second part of the exploratory data analysis will focus on the natural language column `tekst`. Figure 3.4 and Figure 3.5 show the frequency distribution for tokens and characters respectively. The results show that the average token count for visits is relatively small. This corresponds with the early observations made in Section 3.1 about the text data containing mostly keywords.

Figure 3.5: Relative frequency histogram for the character count per appointment

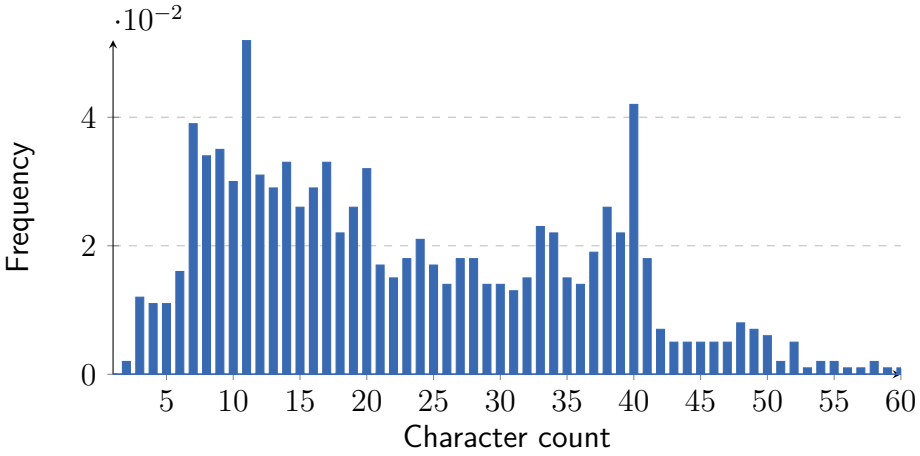


Table 3.4: Overview of the target ICPC codes

	Absolute	Percentage of total
Hypertension	20,799	6.09%
Diabetes	10,759	3.15%
Social issues	3,617	1.06%
Myocardinfarct	1,212	0.35%
Migraine	1,171	0.34%
Stroke	452	0.13%
Pre-eclampsie	1	0.00%
Other	303,453	88.87%

The dataset contains 1700 unique ICPC codes. In order to reduce the dimensionality of the classification problem this research project will focus on classifying a preselected subset of ICPC codes. These ICPC codes where chosen in collaboration with clinical experts from the LUMC. The target ICPC codes are of high interest since in practice they are often under-reported even though they have a large impact on patients, research, and health services. To conclude the exploratory data analysis, Table 3.4 provides an overview of these target diagnoses with their ICPC code and presence in the data. Note that the ICPC code for "pre-eclampsie" (preeclampsia) was labeled as a target ICPC code but it is not sufficiently represented in the dataset used for this research project. Therefore, in the remainder of this project this ICPC code will not be used as target ICPC code.

3.3 Preprocessing

The preprocessing of the data consist of two parts: cleaning and transforming. The first part of the data cleaning is partly described in Section 3.2 where missing values and duplicate data are removed from the dataset. The second part of the cleaning in process involved filtering older instances from the dataset. Figure 3.3 shows that the majority of the data in the dataset is recorded after 2007. As noted by clinical experts from the LUMC, the older data in this dataset is not reliable in terms of the recorded ICPC codes therefore the decision was made to filter out all instances that are recorded before 2007.

The second part of the data cleaning will focus on reducing the noise from the text data. All text data is processed as follows:

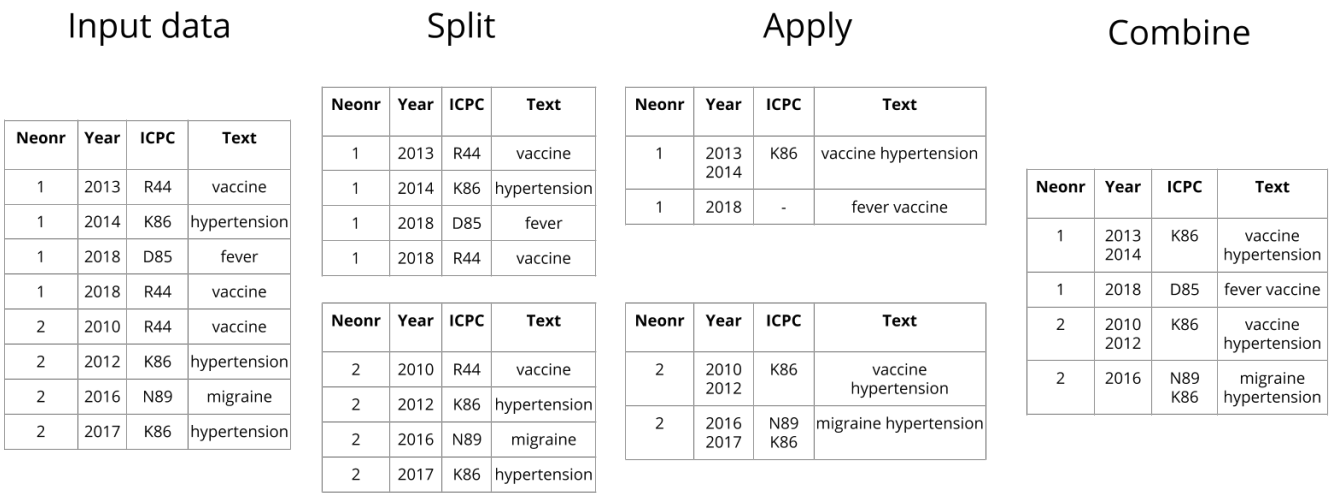
1. All digits are removed. The use of numbers in the text field seem to provide little to no valuable information. In most text instances numbers are only used to refer to a date or used for volumetric units.

2. All special characters are removed as the few special characters in the data are not used for punctuation. Contrary to the previous cleaning step, in this step the removed character is replaced by a white space.

3. All characters are converted to lowercase as the majority of the text is already lower case and capitalization is not used for starting sentences.

It is likely that the digits in the text contain information about the possible diagnosis. For example, the occurrences of "20mg" in the text in combination with the ICPC code for diagnosis might indicate that this volumetric unit is used in the treatment for diabetes. The models can use this extra information to improve

Figure 3.6: The split-apply-combine method visualized



the classification. However since the scalability of the model is one of the requirements, the model should also be developed with the larger subset of target ICPC codes in mind. With more target ICPC codes, or all of them, classifying based on these volumetric units becomes more unreliable given the likelihood of overlapping units without a relation between the diagnoses.

The next step of the preprocessing is data transformation. The transformation technique will follow the well known split-apply-combine mechanism [24]. On a high level, the dataset will be split into separate partitions based on a key. To each of these groups, a custom function is applied which in this case performs a grouping operation within the partition. The last step combines all instances and returns a new dataset. Figure 3.6 visualizes these 3 sequential steps given some (randomized) input data. Note that the "Neonr" column is an unique identifier for patients.

In summary, the split-apply-combine operations in this research projects follow the steps shown in the example and is implemented as follows:

1. **Split** The data is split on NEONR. Each partition represent all visits for a given patient.

2. **Apply** Each partition is grouped by a three year period (e.g. 2012-2015) as requested by clinical experts within the LUMC. As shown in Figure 3.6 in the Apply step, this grouping requires some sort of concatenation of the remaining fields. The text values of all visits that occurred in this three year period are concatenated in one large string whereas the ICPC codes are joined together in a list. In this step the non-target ICPC codes are

also removed. Instances without a target ICPC codes are given an empty list. For demonstration purposes, the Year attribute is also concatenated. In practice however, the Year is derived from the Date field and after the Apply step this attribute is discarded since the models are only trained on the text data.

3. **Combine** Each partition now contains the visits per 3 year period for each patient. In this last step, all partitions are joined together.

Table 3.5: Target ICPC codes overview after preprocessing

	Absolute	Percentage of total
Diabetes	1,082	6.38%
Hypertension	2,625	15.48%
Migraine	237	1.40%
Myocardinfarct	257	1.52%
Social issues	1,409	8.31%
Other	12,083	71.25%
Total	16,958	100.00%

The split-apply-combine operations change the interpretation of an instance in the dataset. Prior to these operations each instance (or row) represented a visit by one patient to the GP which was labeled with the corresponding ICPC code. After the transforming operations, each instance represents a three year period of visits for a given patient. The ICPC label over this time period is the list of all ICPC label occurrences in this time range. Table 3.5 provides an overview of the representation of the target ICPC codes in the processed dataset. At this point, the data is saved as a new copy. Both of the machine learning models (SVM and BERT) will read from the same processed data.

3.4 Training

3.4.1 SVM

Before the SVM is able to perform any type of classification task on this data the text needs to be converted to numerical feature vectors. The "Bag-of-words" approach is the first step in this conversion and consists of two steps [18]:

1. Assign a fixed integer to every distinct token in the corpus. This fixed integer will serve as an unique identifier for the token.

2. For each document i in the dataset X, count the number of occurrences for each token w and store the integer value in $X[i, j]$ where j is the index of the word w in the vocabulary.

Note that the resulting document will only hold the information regarding the tokens and their occurrences. During this step of the training the order of the tokens is lost. Figure 3.4 shows that the average token length of the documents is relatively small in general with outliers with token counts up to 50. The token counts for these document will be greater than average even though they might contain the same information. To scale the feature vectors the occurrences are converted to frequencies using the "Term Frequency" (TF) method. The TF $tf_{t,d}$ of term t in document d is defined as the number of times that t occurs in d. This will scale the originally high occurrences to normalized frequency values. Furthermore, rare terms that might contain valuable information are overshadowed by more common words (e.g. stop words). The "Inverse Document Frequency" weights down common words that carry little valuable information. The document frequency df_t of term t is defined as the number of documents that contain t. Given the entire corpus N, df_t/N can be seen as the likelihood of the occurrence of t in any document. The inverse document frequency quantifies the probability of term t in a document and is defined as follows:

$$idf_{(t)} = \log(\frac{N}{df_{(t)}}) + 1$$

To conclude the weighting scheme, the final tf-idf weight is the product of both the TF and IDF weights:

$$tf\text{-}idf_{(t,d)} = tf_{(t,d)} \cdot idf_{(t)}$$

Including the SVM model training, the training process follows three steps which are compounded into one pipeline:

1. Vectorizing (Bag-of-words)
2. Transforming (Tf-Idf)
3. Classifying (SVM)

The SVM training concludes with a hyper parameter experiment. The `sklearn` library allows for optimizing the parameters for each step in the pipeline. Since the count vectorizer and the tf-idf transformer are also implemented in this pipeline, these parameters can also be optimized using a single parameter grid. The `GridSearchCV` method performs an exhaustive search over all possible parameter combinations and cross-validates the results. For this experiment, the following parameters are used:

- **N-gram range (count vectorizer)** By default, the `ngram_range` is set to 1 which implies that a single token is converted to a column. For some use cases it can be beneficial to also evaluate the performance of bigrams which consists of two words. In this case the combination of the two unigrams "blood" and "pressure" are combined in to a single bigram "blood pressure".
- **Use idf (tf-idf transformer)** This flag controls the use of the inverse document frequency during the tf-idf method.
- **C parameter (SVM)** The C parameter tells the SVM optimization to what degree classifications are penalized. The SVM example in Figure 2.1 visualized the decision boundaries by the SVM algorithm. A high C value would lead to a smaller margin for the support vectors and vice-versa.

One of the dangers of this hyper parameter optimization is overfitting on the test data. When this is the case, the parameters will be optimized to perform best on the given test data which is undesirable as these performance increases will not translate to new, unseen data. To validate the performance differences the hyper parameter optimization will be cross-validated on the training data only. While this prevents overfitting, it also allows for using one test dataset for both models which will result in a more consistent comparison.

Table 3.6: The hyperparameter grid for the SVM

Hyperparameter	Default value	Gridsearch range
ngram_range	(1,1)	[(1, 1), (1, 2)]
use_idf	True	[True, False]
C parameter	1	[0.1, 0.5, 1, 1.5, 2]

Table 3.6 provides the complete overview of the used hyperparameters, the default value used for the baseline model and their corresponding ranges for the optimization. The sklearn class `GridSearchCV` provides a grid search which exhaustively generates candidates from a grid of parameter values. In order to prevent the previously described risk of overfitting the method also cross-validates the training data using a 5-fold cross-validation generator.

The background information of the SVM in Subsection 2.2.1 demonstrates the basic mechanics of a SVM using a binary classification problem. However, the SVM can also be applied for a multiclass classification problem as is the case in this research project. Given a classification problem with N classes, there are two approaches for training the SVM:

- **"One-vs-One"** A binary classifier is trained for every combination of two classes in N. Each classifier is trained using only the samples from the two selected classes. While predicting, a voting function assigns labels to test data. This methods results in $\frac{N(N-1)}{2}$ classifiers.
- **"One-vs-Rest"** A binary classifier is trained for every class in N with the samples of the selected class and all other samples. Each classifier is trained to distinguish the selected class with all other classes. During testing, test data is labeled with the selected class if the binary classifier in question returns this class and not the "rest". In this method only N classifiers are used.

In this project, the Python package `scikit-learn` (`sklearn`) is used for implementing the SVM. In the built-in SVM class, multi-label support is handled according to the One-vs-Rest scheme [19].

3.4.2 BERT

The implementation for the BERT model is based on the `Transformers` library developed by the Huggingface team [25]. The library provides a unified API for handling NLP tasks such as configuration, tokenization and the use of pre-trained models. This unified API allows for smooth interchangeability between different pre-trained models, tokenizers and configurations which is beneficial for experimenting with different setups in this research project. The `Transformers` library is deeply compatible with the two popular deep learning frameworks PyTorch and Tensorflow (from version 2.0). This improves the transition for models from research to a production environment as the models also support `TorchScript` which is able to serialize and optimize models for PyTorch which can be used to deploy or incorporate models in to a production environment using the Tensorflow Extended framework.

One of the powerful features of the `Transformers` library is the use of pre-trained language models. As explained in more detail in Chapter 2 these models are trained on a large corpus of text and can be fine-tuned for downstream tasks. This research project will utilize this feature by using a pre-trained model and fine-tuning it with the classification tasks of labeling GP notes with an ICPC code. The pre-trained language models are mostly language specific with the exception of the multilingual model which consists of the top 100 languages with the largest Wikipedias as measured by the number of articles and includes Dutch. To account for the difference in Wikipedia sizes, exponentially smoothed weighting of the data during pre-training was performed. This results in high-resource languages such as English to be under-sampled while low-resource language such as Icelandic were

over-sampled. Intuitively it seems inefficient to apply a language model containing 100 language to a Dutch classification problem. However, experiments have shown that the results are similar to Dutch language models used on Dutch training data [23]. In order to speed up the development process, and since the results are very similar, this research project will only use a Dutch BERT language model.

BERTje is a Dutch pre-trained BERT model developed at the University of Groningen [23] and will be used in this project. While the multilingual models training data is limited to Wikipedia, BERTje used data from other sources including the Dutch Wikipedia. The corpora used for pre-training the BERTje model are as follows:

- Books: a collection of contemporary and historical fiction novels
- TwNC: a Dutch news corpus
- SoNaR-500: a multi-genre reference corpus
- Web news: all articles of four Dutch news websites from January 1, 2015 to October 1, 2019
- Wikipedia: the October 2019 dump

The implementation of the BERT model is done using the `simple-transformers` package which is a wrapper around the original Huggingface Python package for transformers [15]. The wrapper allows for faster development while also providing essential hyperparameter options to configure the model. As for the BERT model, the following hyperparameters were used in the optimization process:

- **Training batch size** Defines the number of instances to be propagated through the neural network in each batch.
- **Number of epochs** Defines the number of complete passes (of the entire training set) through the neural network
- **Learning rate** Controls the weights adjustments of the neural network after each batch.

Table 3.7 provides the complete overview of the used hyperparameters and their corresponding range. Note that the implementation used for evaluating the parameters resembles the implementation of the sklearn `GridSearchCV` class where the approach uses an exhaustive (or brute force) search to generate and evaluate all possible combinations of the parameters.

Table 3.7: The hyperparameter grid for the BERT model

Hyperparameter	Range
train_batch_size	[8 , 16, 32, 64]
learning_rate	[1e-3, 1e-4 , 1e-5, 1e-6]
num_train_epochs	[1, 2]

3.5 Evaluation & Metrics

3.5.1 Metrics

The goal of this research project is to develop a model that is able to correctly predict the ICPC codes given a sample of GP visits. Evaluating the models is an important step in the machine learning process. It is important to select the "best" model during training in order to get the most information of the training data while preventing overfitting on the training data which leads to the model not being able to generalize well on unseen data. There are several techniques for evaluating the performance of the model and most of these metrics can be used in conjunction to provide a more general overview of the model performance. Most of the metrics used are based on true positives, false positives, true negatives and false negatives:

- A true positive (TP) is an outcome where the model correctly predicts the positive class. A true negative (TN) is an outcome where the model correctly predicts the negative class.
- A false positive (FP) is an outcome where the model incorrectly predicts the positive class and the false negative (FN) is, subsequently, the outcome where the model incorrectly predicts the negative class.

The examples above describes a binary classification problem with a positive and a negative class. In this research project the model is trained for a multi-label classification problem with N classes. In this case the four prediction outcomes (TP, TN, FP, FN) are calculated separately for each class and averaged. During evaluation, the prediction outcomes can be summarized by different metrics. In the next subsection the most relevant metrics for the research project will be described.

Accuracy

The most popular (and most intuitive) metric used for evaluating machine learning models is accuracy. Accuracy defined as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

In other words, accuracy is the fraction of correctly classified instances over the total amount of instances. While intuitive, a high accuracy does not automatically imply a well performing model. For example, given a model trained to classify spam (from non-spam) email where the testing data consist of 9.000 non spam email and 1.000 spam emails. A model which labels every test instance as a non-spam email will achieve an accuracy score of 90% while the model fails to perform the indented task and is useless in practice. This example shows that accuracy does not always correctly reflect the performance of classifiers when working with unbalanced data. Therefore, it is preferable to use precision and recall instead of accuracy.

Precision, Recall and F_1 score

Precision is a metric that defines the fraction of true positive classes over the total amount of positive classes. In other words, what portion of the positive identifications were actually correct. Given the evaluation of a spam classifier, the precision is high when the majority of the emails classified as spam, were actually spam emails. The precision metric is defined as follows:

$$Precision = \frac{TP}{TP+FP}$$

The recall metric on the other hand attempts to answer what proportion of the actual positives classes was identified correctly. Again using the spam email classifier as an example, the recall is high when the majority of the actual spam emails, were correctly identified as spam. In other words the recall reflects how many of the actual spam emails the classifier was able to catch. The recall metric is defined as follows:

$$Recall = \frac{TP}{TP+FN}$$

Lastly, the F_1 score (also knows as F-score of F-measure) can be interpreted as the weighted average of the precision and recall. More specifically, the F_1 is the *harmonic mean* of the precision and recall. Instead of computing the regular means the F_1 score gives more weight to low values from either the precision or the recall. Therefore in order to het a high F_1 score, both the recall and precision need to be high. The F_1 score is less intuitive than the precision and recall metric but for completeness sake this metric will also be reported in this research project. The F_1 score is defined as follows:

$$F_1 = \frac{recall^{-1} + precision^{-1}}{2} = 2 \cdot \frac{recall \cdot precision}{recall + precision}$$

To fully evaluate the performance of a model both the precision and recall have to be examined. However, the precision and recall are often in tension meaning that increasing one is likely to decrease the other. Choosing the right balance between these two metrics depends on the nature of the classification problem. Given the spam email classifier working in a production environment, incorrectly classifying email as non-spam in undesirable but it the impact is relatively small. Contrarily, classifying a non-spam email as spam has a big impact since this email will be placed in the spam folder and missed by the user. Therefore in this example, the precision is more important than the recall. On the other hand, for a classifier trained to detect shop lifters from video surveillance the recall should be optimized since catching most shoplifters is more important than the accuracy (precision) of shoplifting instances.

3.5.2 Train-test split

For both models, the dataset is split according to the 80:20 train-test split ratio where 80% of the data is used to train the model while the other 20% is used for testing. The same training and testing data is used for both models to ensure consistent results. As explained in Section 4.2, the BERT model uses a variation on this principle where 20% of the training data will be used for validating the model during training resulting in a slightly smaller training set while keeping the testing set consistent between the models. For optimizing the hyperparameters of the SVM model, the same principle is used where 20% of the training data is used for finding the most optimal hyperparameters while the resulting model is tested on the original test set.

Chapter 4

Results

In this section the results of both machine learning algorithms will be evaluated and compared. Both models will consume the same preprocessed dataset exported from Chapter 3 and will use this as the starting point for training, while further model specific preprocessing will be explained in the corresponding sections.

4.1 SVM

The first objective of the SVM evaluation is to develop a baseline model. This model will be trained and evaluated on the preprocessed dataset without further configurations to the options and hyperparameters of the model. Contrary to the data preparation of the BERT model, the preprocessed data will be vectorized and transformed using the count vectorizer and the tf-idf transformer before the training process starts.

Table 4.1: The classification report for the SVM classifier

	Precision	Recall	F1-score	Support
Diabetes	0.935	0.642	0.761	226
Hypertension	0.917	0.726	0.810	533
Migraine	0.788	0.605	0.684	43
Myocardinfarct	1.000	0.415	0.587	53
Social issues	0.850	0.526	0.650	270
micro avg	0.904	0.642	0.751	1,125
macro avg	0.998	0.583	0.698	1,125

Table 4.1 provides the classification report for the baseline SVM model. The report shows that the precision of all target ICPC codes is relatively high which

indicates that the false positive rate of the model low. Compared to precision, the model performs worse when evaluating recall. In terms of recall, the model performs best for the target ICPC code Hypertension where approximately 75% of all instances are correctly predicted by the model whereas the model is only able to predict less than half of all the Myocardinfarct instances in the test set.

Table 4.2: Top 20 SVM feature coefficients

	Diabetes	Hypertension	Migraine	Myocardinfarct	Social issues
1	metformine	hypertensie	migraine	myocardinfarct	partner
2	mellitus	hydrochloorthiazid	imigran	coronairsclerose	relatieproblemen
3	gliclazide	hypertens	flikkerscotoom	coronair	burn
4	glimepiride	amlodipine	zolmitriptan	infarct	zorgen
5	hbac	losartan	smelttabl	nstemi	problemen
6	diabetes	beschad	rizatriptan	onderwandinfarct	echtgenoot
7	type	chloortalidon	sumat	hartinfarct	arbeidsconflict
8	gereguleerd	cozaar	zomig	cabg	moeder
9	insuline	hydrochl	propranolol	stent	relatieprobleem
10	behandelplan	losart	orod	acuut	relatieproblematiek
11	dmii	metoprololsucc	neurapraxie	myocard	werksituatie
12	tolbutamide	valsartan	oxynorm	stemi	vader
13	solos	hydrochloorthiazide	reinigen	vatslijden	dochter
14	lantus	orgaanbeschadiging	endocarditisprofylaxe	omeprazolgebruik	werk
15	diabetische	enalapril	ftab	voorwandinfarct	ontslag
16	afhankelijke	triamter	cluster	ischemische	probleem
17	glycohb	perindopril	dingen	mell	relatie
18	diab	lisinopril	aura	pravastatinenatr	overlijden
19	jaarcontrole	hydroch	endometriumcarcinoom	coronairlijden	ouders
20	flexpen	tensie	oxycodon	lijden	werkproblemen

The SVM is a linear classifier where its features are represented by all the tokens in the corpus of document. One of the options of the SVM is the ability to rank these features by relevance for the prediction of a certain ICPC code. In other words, this makes it possible to print a list of words which are most informative for each class. Table 4.2 shows the top-20 features for all target ICPC codes in order to provide a more visual insight in the decision making process of the SVM model. For most cases the most informative features are the diagnosis name followed by related medical prescriptions. Examples are "Tolbutamide" which is a potassium channel blocker used for the management of diabetes type 2 or "Chloortalidon" which is used to treat high blood pressure. Besides the various medicine related to the diagnosis there are also terms which seem out of place as highly informative feature

coefficients. Examples are "afhankelijke" (dependent), "dingen" (things) and "lijden" (suffering) which may indicate that the model is overfitting on the training data. Lastly, the table also shows multiple occurrences of the same concept as different features. An example of this can be found for the Social issues diagnosis were "Relatieproblemen" (relationship problems), "Relatieprobleem" (relationship problem), "Relatieproblematiek" (relationship problems) and even the conjunction of "Relatie" (relation) and "Probleem" (problem) all refer to the same concept while being separated over 5 different features.

Table 4.3: The SVM results before and after the hyperparameter optimization

	Precision	Recall	F1-score	Support
Before optimization	0.904	0.642	0.751	1,125
After optimization	0.887	0.675	0.766	1,125

In the final experiment the effect of hyperparameter optimization is evaluated and compared with the baseline model. The `sklearn` SVM class allows for configuring the regularization parameter and loss function the algorithm uses during training. In addition to optimizing the SVM both the count vectorizer and tf-idf transformers can also be optimized in the same process. The most relevant parameter for the count vectorizer is the n-gram range which enables the count vectorizer to treat multiple words as one feature. Given the parameter n, $n = 1$ are unigrams, $n = 2$ bigrams and so forth. During this experiment $1 \leq n \leq 4$ is used during the hyperparameter optimization. Table 4.3 shows that the hyper-parameter optimization leads to a minimal increase of the recall and f1-scores at a slight costs of precision.

The classification report of Table 4.1 reports the precision and recall of the target ICPC codes. Figure 4.1 shows the confusion matrix results of the (optimized) SVM model providing better insight in what happens in the cases where the model is wrong. When observing the columns left to right the results show that when the model predicts one on the target ICPC codes it generally aligns with the actual ICPC label. For the majority of the classes in the case the model assigned a wrong ICPC label it was the "Other" class. Only for the Hypertension class it can be seen that in a small amount of cases the model wrongly assigned instances with the Diabetes class. Similar results can be observed when evaluating the recall represented by the rows in the figure. For the target ICPC codes, cases which are not correctly predicted are generally assigned with the "Other" class.

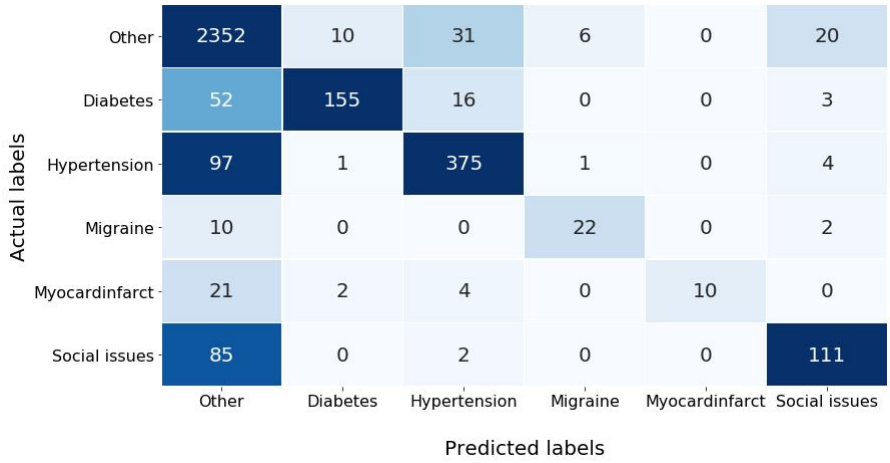


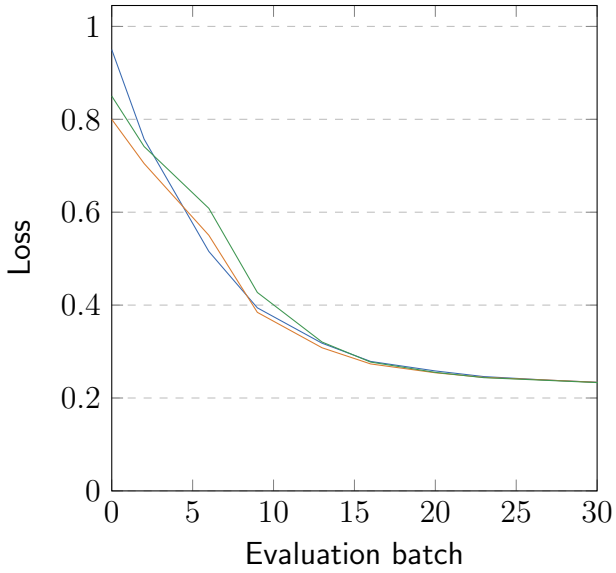
Figure 4.1: The confusion matrix for the SVM model

4.2 BERT

The first experiment, the hyper parameter optimization, attempts to find the optimal combination of hyperparameters for training the BERT model. For the BERT model, it was slightly more difficult to start with a baseline model as the `simple-transformers` library does not provide default parameter suitable for this project. Therefore instead of starting with a baseline model this experiment starts with the hyperparameter optimization. Initially, the learning rate, training batch size and the number of epochs were labeled as the target parameters used in the optimization process. One of the challenges encountered during this experiment was the increased training time for multiple epochs. To illustrate, each epoch takes roughly 10 minutes to complete. Given a parameter grid with only 4 values for each of the 3 target parameters, and given the fact that the majority of the runs would then consist of multiple epochs, this experiment would take a couple hours to complete. Furthermore, the experiments are conducted via a remote desktop client in order to work within the LUMC environment which (by default) features automatic timeouts and disconnects making longer (or overnight) runs difficult. Lastly, the memory of the GPU occasionally failed to clear in between runs which required a restart of the notebook kernel. In practice the combination of these challenges causes this experiment to fail at every attempt. Therefore the decision was made to remove the number of epochs parameter from the experiment and evaluate it manually by running 2 experiments for different epoch values. This manual experiment confirmed that the number of epochs after 1 had little to no effect on the performance of the model.

Contrary to the SVM, the BERT model is a deep neural network which requires training on the training data while keeping track of the loss metric in order to

Figure 4.2: The loss of the BERT model for three separate runs



move to a minimum in which the network is able to converge to the point were the returned loss metrics does not decrease any further. The objective of the second experiment is to confirm the model is able to reach this minimum in which there is no more improvement possible given the provided data and configuration options. Before this minimum is reached the network is in theory able to learn more and perform better therefore skewing the results of the model comparison in this research project. Figure 4.2 visualizes the loss calculated over the validation set given 30 evaluation batch steps. Due to the nature of neural networks and their random weight initialization, this experiment was repeated several times to ensure consistent results. The figure shows that in all three experiment runs the network starts at slightly different initial losses but is then able to converge to a minimum in which the loss metric stops decreasing.

Table 4.4 provides the classification report for the BERT model with the optimal parameters as reported in the previous experiment. The report shows that the model is not able to learn the two ICPC codes with the lowest support. Both "Migraine" and "Myocardinfarct" have only 50 instances in the test set. Given the 80:20 train test split ratio with an additional 20% of the training set reserved for validation, this amounts to approximately 150 instances per ICPC code being used during training which seems to be insufficient for the neural network to properly train. Figure 4.3 provides the confusion matrix of the BERT model. The confusion matrix provides more insight in the misclassified instances. The results show that in the majority of the misclassified instances the model predicts the "Other" category.

Table 4.4: The classification report BERT classifier

	Precision	Recall	F1-score	Support
Diabetes	0.874	0.491	0.629	226
Hypertension	0.952	0.638	0.764	533
Migraine	0.000	0.000	0.000	43
Myocardinfarct	0.000	0.000	0.000	53
Social issues	0.840	0.389	0.532	270
micro avg	0.913	0.494	0.641	1,125
macro avg	0.533	0.304	0.385	1,125

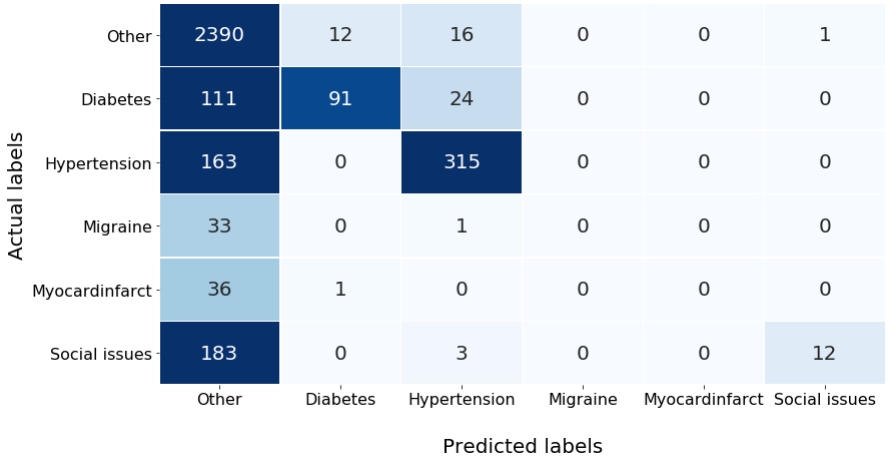


Figure 4.3: The confusion matrix for the BERT model

For the larger target ICPC codes (Diabetes, Hypertension, Social issues) the models performs well in terms of the precision which ranges from 75% to 90%. The recall however shows that the model struggles to "catch" all of the instances with target ICPC codes as the recall is at 62% on average. Both metrics generate and average F1-score of 72%.

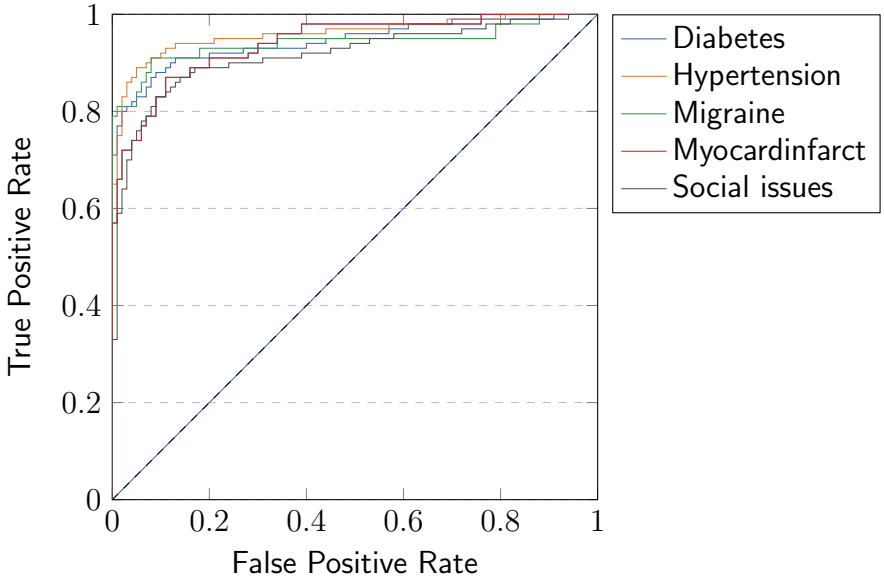
4.3 Comparison

Table 4.5 provides the results for both models with the micro and the macro metric averages. The macro average is the unweighted average of both the precision and recall of all classes and the BERT models scores substantially worse than the SVM since the model was unable to predict two of the five classes. For the micro average, the two models score relatively similar in terms of the precision

Table 4.5: The precision, recall and f1-score of the SVM compared with the BERT model

		Precision	Recall	F1-score	Support
Micro	SVM	0.877	0.675	0.766	1,125
	BERT	0.900	0.444	0.594	1,125
Macro	SVM	0.889	0.585	0.688	1,125
	BERT	0.529	0.254	0.321	1,125

Figure 4.4: AUC/ROC curve for the SVM model



but overall the SVM model outperforms the BERT model in every other measured aspect of this experiment.

The importance of the precision compared to the recall varies between different applications for machine learning models. The main problem which led to the development of the models in this research project is the inability to conduct research based on the recorded ICPC codes in the past. Therefore one of the possible applications of a model is annotating historic data in the medical records. When evaluating the trade-off between precision and recall in this scenario the precision is arguably more important than the recall. When annotating missing data every correctly predicted ICPC code is an improvement over the initial situation, whereas each incorrect prediction deteriorates the quality of the medical records even further.

Table 4.6: Time comparison between training on the training set and predicting on the data with missing ICPC codes

	SVM	BERT
Annotating missing ICPC codes	8.07s	331.34s

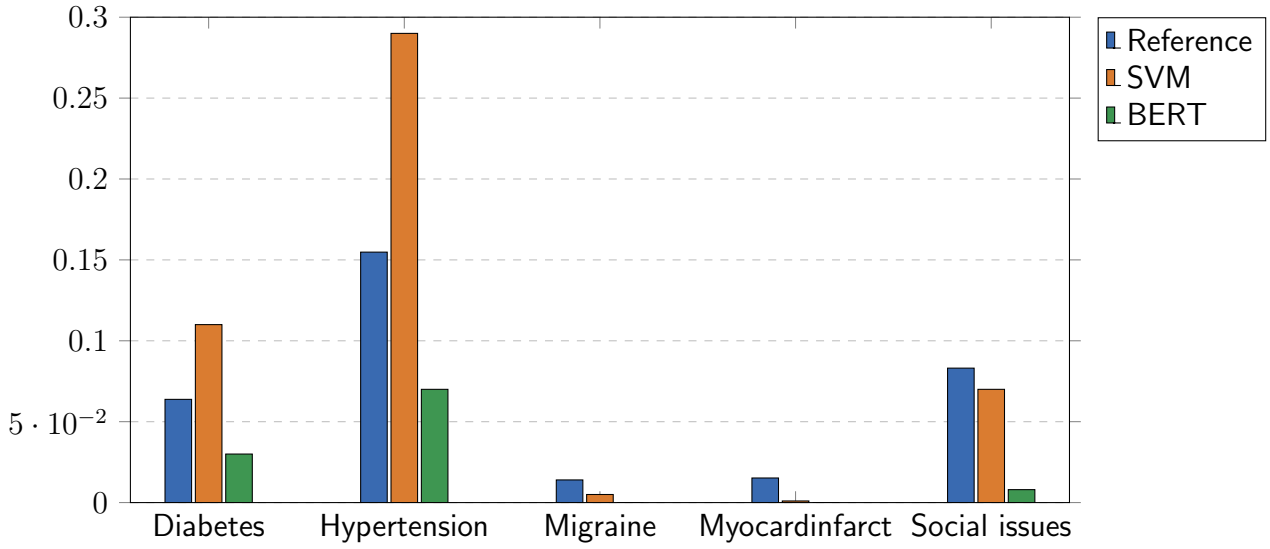
One of the causes of the under-reporting of the ICPC codes in the medical records as identified by clinical experts is the time-pressure GPs experience during their work. Therefore a different application would be a suggestion or recommendation engine built into their routine. That is, after entering the free text an application would read the text and suggest the most likely diagnoses. In this scenario predictions would be presented as suggestions and it is up to the GP to evaluate and assign the correct ICPC code. A wrong prediction leads to one or more suggested diagnoses which are unrelated to the presented case. This however only has a negative impact if the GP assigns this label. In contrary, systematically not suggesting the correct diagnosis, which is the case with a low recall, leads to the same under-reporting problem due to time-pressure. In this case the recall is expected to be more important compared to the recall.

4.4 Labeling the unlabeled data

The exploratory data analysis of Section 3.2 describes how roughly 90% of the data is not labeled with a ICPC code and therefore can not be used for training the models in this project. In the last part of this section the trained models on this unlabeled data. As described in Section 3.3, the instances in the dataset are not trained separately but instead are grouped by patient and in intervals of 3 years. To ensure consistent results, the same preprocessing steps are applied to this unlabeled dataset.

Table 4.6 shows the evaluation time of both models.This evaluation time corresponds to the time it takes for a single model to make predictions on the remaining unlabeled data. This value therefore excludes preprocessing and grouping as the steps are the same for both models. Due to the random weight initialization of the BERT model the value for this model is the average of 3 runs. Lastly, all experiments are conducted on the LUMC provided virtual machine. This means the values should be evaluated relative to each other since the actual time in seconds can vary between machines or servers. The results shows that the SVM completes the task in 8 seconds while the BERT model takes 5 minutes and 30 seconds to

Figure 4.5: Target ICPC codes distribution on the unlabeled data compared to the training data (reference)



complete making the SVM model almost 40 times faster.
Since the remaining data is unlabeled it is not possible to evaluate the performance metrics as could be done with the training data in the preceding result sections. What can be done is visualize the distribution of the labels on this unlabeled dataset in order to compare them to the distribution of the labeled dataset. Figure 4.5 shows the distribution of the target ICPC codes for both the SVM model and BERT model. In the figure, these values are compared to the training data which is labeled as reference in the figure. For the BERT model, the results align with the results reported in Section 4.4. The model is unable to predict both Migraine and Myocardinfarct due to their low support during training while it is able to predict roughly half of the Diabetes and Hypertension instances. Contrary to the previously reported results, the BERT model also seems to under report the Social issues ICPC code.

Chapter 5

Discussion

The final results of Table 4.5 show that both models score well in this classification task in terms of their precision while both models struggle with the recall. Contrary to other types of multi-label classification problems the model in this project is allowed to return an empty prediction. In this case the presented text does not fit with any of the target ICPC codes which explains the imbalance between the precision and recall. For the BERT model the low recall performance drags the F1-score down to the point where it can not compete with the SVM model. The low recall metric reflect the inability of models to "catch" all target ICPC codes but it does not provide the cause of this imbalance between the precision and the recall. The confusion matrices for both models in Figure 4.1 and Figure 4.3 show not only the high precision but also that the misclassified examples are mostly labeled with "Other". In a use case where the "Other" label is treated as unknown, and every correctly classified instance provides business value in its own, both models can be used to improve the annotation of the data. However, the lower recall implies that the models struggle to catch all target ICPC codes which means the annotated data would not give an accurate overview of the epidemiological environment.

Table 5.1 highlights some instances in the test dataset where both models failed to predict the target ICPC code(s) but instead predicted "Other". Note that the data in the table is a very small sample of handpicked instances with the two most important criteria being that the text does not include any potential privacy sensitive information (hence Social issues is excluded) and relatively short text was used for clarity. Two observations can be made from the limited amount of data presented in table 5.1. First, the data from the table, and a manual inspection, show that in some cases the labeled ICPC code does not align with the corresponding text. Given the problem statement of this project however, which notes the incompleteness and the current inability to verify ICPC codes, this is not surprising. The second observation is the lack of natural language in the text

Table 5.1: Sample of instances both models failed to predict the target ICPC code but instead predicted "other"

Text	**Actual label**
andere tendovaginitis tendinitis knie wratjes voetklachten otitis externa pravastatine tabl kenacort bursitis subacromialis	Hypertension
zwelling knie griepvaccinatie batchnr oordopje gehoorgang verhoogd atenolol tabl prostaatcarcinoom tanmx kattanscore griepvaccinatie gegeven batch	Hypertension
selokeen tabl	Diabetes
bovenste luchtweg infectie dieetiste vind niet nodig gaat naar oogarts contr glaucoom diabetes mellitus type niddm ooglidklachten amoxicilline caps otitis otitis media codeine phosphas tabl domperidon zetp codeine tabl zolpidemtart tabl slapeloosheid andere slaapstoornis	Diabetes, Migraine
bloedond veneuze insufficiantie lichte polyneuropathie werkdiagnose mogelijk lichte arthrose ezetimib gewijzigd medicatie historie psoriasis arthrose pols cellulitis secundair geinfecteerd krabeffecten cave droge otitis externa werkdiagnose lwinf sinusitis tevens bronchitis myalgie bijwerking ascal status	Hypertension
verrucae seborroica amoxicilline caps ventolin diskus parestesie restklachten dyspnoe affort pijn teen prednisolon tabl allopurinol tabl lanol gycerine neusza droge neus tramadol caps ischias oxycodon caps oxycodon tabl colchicine tabl colch jicht furabid caps nortriptyline tabl	Diabetes
reactie viraal infect chalazion atorvastatine tabl voetcontrole oproep juli pijn bovenbuik hydrochloorthiazid ofloxacine oogdr amoxicilline caps zure oordr triam meroc oortamp odta flucloxacilline caps miconazol oordruppel haematoom been zweten oproep eerst bloed urineonderzoek schouderklachten snijwond benauwd borst	Diabetes, Hypertension
verruca seborroica griep redelijk tractus iliotibialis excisie weer verwijderd ribcontusie spastisch colon griepcampagne	Hypertension
schouderklacht links griepvaccinatie gegeven batch nagel probleem duim artrose handen verruca seborroica atorvastatine tabl vergeetachtigheid griepvaccinatie lotnummer	Myocardial infarction

field. This is partly due to the grouping process during preprocessing but it also applies to single instances which shows that GPs in most cases fall back to using keywords instead of full grammatically correct sentences. As described in-depth in Section 2.2.2, the main advantage of the BERT model stems from the attention mechanism which enables the model to defer the meaning from a words from the words around it in the same sentence, similar to how humans interpret words. This advantage however is mostly wasted due to the absence of natural language in the training text. This characteristic of the training text might explain the performance differences between both models as the SVM model is based on the bag-of-words model and therefore does not require sentence structure and word order.

The BERT model faces several challenges compared to the SVM model. Table 4.4 show that the model is unable to predict two of the five classes in the dataset used for this project. The Migraine and Myocardifarct class have roughly 150 training instances each which seems to be an insufficient amount for the model to properly train and at this point in the research it is also unclear if this problem would have been resolved with more instances. Regardless of the size of the training data, it does show that the model struggles with under-supported classes which is likely to occur when the target ICPC codes are expanded in future research. Secondly, the BERT model requires significant computational power compared to SVM. The fine-tuning task requires a specialized GPU as consumer grade CPUs (and even consumer grade GPUs) are not capable of performing this task in reasonable time. This places several constraints on the flexibility and scalability of this model. Given the size of the dataset used in this project, system requirements for training the SVM model are low enough for average consumer laptops to run comfortably. The lack of the GPU requirement significantly improves development time and overall (development) costs as it makes it easier to continuously improve the model by running experiments with more data, better feature engineering or a wider subset of target ICPC codes. Lastly, Table 4.5 also shows that the evaluation time, the time it takes for the model to perform predictions, is roughly 40 times slower then the SVM model when performed on the remaining ICPC codes. Depending on the use case but especially where batch predictions are performed this could also prove to be a disadvantage of the model.

Table 2.1 from the prior work analysis showed that for clinical NLP tasks the BERT models trained on domain specific tasks generally outperform general BERT models however at this time of writing there are no public biomedical BERT models in Dutch available. Pre-training a BERT model requires a large corpus of text and significantly more computational power than the fine-tuning tasks. Given the scope and time of this research project, both collecting a large biomedical corpus and pre-training the BERT model were not realistic. However, given the similarity

of the both models in this task, and the differences of the domain specific models this does provide interesting future work.

In the evaluation of the SVM performance the observation was made that the top coefficients of the SVM model in Table 4.2 show multiple occurrences of the same concept as different features such as "Relatieproblemen" (relationship problems), "Relatieprobleem" (relationship problem), "Relatieproblematiek" (relationship problems) existed. These features all refer to the same concept and should ideally be combined. In this case techniques such as stemming, lemmatization but also regular text correction can be used in order to reduce the dimensionality of the feature vectors. In this research project however, no suitable implementation could be found which works on Dutch biomedical text data and developing such solution from scratch was outside the scope of this project. Fortunately, the target ICPC codes used are unlikely to have much overlap in the features as the diagnosis are not closely related en Table 4.2 which provides the overview of the most informative features support this hypothesis. For further research however with more data, and especially when working with more target ICPC codes which might share some overlap in features, these type of techniques can could benefit the performance of the model.

Subsection 3.5.2 describes the train-test split method used in this project. The initial data (suitable for training the classifier) is split randomly in a 80%-20% ratio. At this point the data is already preprocessed which included cleaning the data but also the transforming and grouping process described in Section 3.3. After transforming, each record the models will be trained on a combination of a patients visits spanning a three year period. Given the follow up time and total visits per patient as seen in Figure 3.2 and Figure 3.3 respectively, the visits of the patients are likely to span multiple blocks of three years. The danger of the random split of the data is that these blocks per patient can fall into both the training data and test data, therefore potentially "leaking" data between the datasets. For example, given a patient with a rare description of a (target) diagnosis not found in the data of other patients, it is possible for this text to fall in both the training data and test data. For classifiers it is now possible to train on this training data and accurately predict the instances in the test data as the classifier has already been trained on this part of the test data. In order to develop a model which is able to generalize better the data should have been split on patient level preventing potential leakage between the training and test set. This would lead to a more accurate and reliable view of the model performance on real-world and unseen data.

Chapter 6

Conclusions

The research question as defined in Chapter 1 consists of two sub questions. The first part of the research question is formulated as follows:

"To what degree can machine learning algorithms predict ICPC encoded diagnoses based on free-text notes"

The capability to predict is a combination of both the precision and the recall. Both models share the same characteristic in this aspect as the precision is generally very high with the exception of the macro precision of the BERT model. As this is an weighted average of all 5 target classes, this is negatively affected by the models inability to predict 2 of the 5 classes and at this point in the research it is unclear if more training data would resolve this issue. The SVM model, at least given the dataset in this research project, has little to no problems correctly predicting under supported classes and therefore reports high micro and macro precision scores. Compared to the precision, the recall of both models is significantly lower and while the importance of the recall generally depends on the use case, the 0.44 (micro) recall of the BERT model is likely to hinder practical use. Given the high precision of both models, and the acceptable recall of the SVM model, this research project has shown that machine learning algorithms are capable of predicting ICPC encoded diagnoses based on free text.

Finally, the second part of the research question:

"How do traditional techniques compare to pre-trained language models"

In terms of the performance of both models as measured by the precision, recall and F1-score, the SVM model outperforms the BERT model in every aspect. The lack of natural language in the training data might be the cause of this disparity as understanding natural language is one of the key improvements BERT introduces compared to other techniques and algorithms. Performance is only one side of the story, the discussion in the Chapter 5 elaborated further on the different trade-offs

between the two models. The BERT model requires significant computational resources and specialized hardware in order to train the model. This requirement severely hinders the flexibility of the model as further development, with more data or better feature engineering, requires more resources and costs. Therefore given the problem as presented in this research project, the research has shown that when comparing SVMs with BERT models, the SVM is the favorable algorithm for classifying the free-text notes of GPs.

Bibliography

[1] Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. Publicly available clinical BERT embeddings. *CoRR*, abs/1904.03323, 2019.

[2] Sören Brage, Bent Guttorm Bentsen, Tor Bjerkedal, Jan F Nygård, and Gunnar Tellnes. ICPC as a standard classification in Norway. *Family Practice*, 13(4), 08 1996.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[4] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[5] Vijay N Garla and Cynthia Brandt. Ontology-guided feature engineering for clinical text classification. *Journal of biomedical informatics*, 45(5):992–998, 2012.

[6] Santiago González-Carvajal and Eduardo C. Garrido-Merchán. Comparing BERT against traditional machine learning text classification. 05 2020.

[7] Google. Understanding searches better than ever before, 2019.

[8] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[9] Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3:160035, 2016.

[10] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language

representation model for biomedical text mining. *Bioinformatics*, 36(4), 09 2019.

[11] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

[12] Bernardo Magnini, Alberto Lavelli, and Simone Magnolini. Comparing machine learning and deep learning approaches on NLP tasks for the Italian language. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2110–2119, Marseille, France, May 2020. European Language Resources Association.

[13] Vincent Menger, Floor Scheepers, and Marco Spruit. Comparing deep learning and classical machine learning approaches for predicting inpatient violence incidents from clinical text. *Applied Sciences*, 8(6):981, 2018.

[14] Mei-Sing Ong, Farah Magrabi, and Enrico Coiera. Automated categorisation of clinical incident reports using statistical text classification. *Quality and Safety in Health Care*, 19(6):e55–e55, 2010.

[15] Thilina Rajapakse. Simple transformers, 2020.

[16] Jason DM Rennie and Ryan Rifkin. Improving multiclass text classification with the support vector machine. 2001.

[17] Thomas Brox Røst, Øystein Nytrø, and Anders Grimsmo. Classifying encounter notes in the primary care patient record. pages 1–5, 2006.

[18] Scikit-learn. Extracting features from text files, 2017.

[19] Scikit-learn. sklearn.svm.linearsvc, 2019.

[20] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[22] Marc Verbeke, Diego Schrans, Sven Deroose, and Jan Maeseneer. The international classification of primary care (ICPC-2): an essential tool in the EPR of the GP. *Studies in health technology and informatics*, 124:809–14, 02 2006.

[23] Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. BERTje: A Dutch BERT Model. *arXiv:1912.09582 [cs]*, December 2019.

[24] Hadley Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software, Articles*, 40(1):1–29, 2011.

[25] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.