Internal Report 2010–09 August 2010

Universiteit Leiden

Opleiding Informatica

Automated Document Classification of Medical Reports

Ruben van Bodegom

MASTER'S THESIS

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

Master's thesis

Automated document classification of medical reports

Leiden Institute of Advanced Computer Science
(LIACS)

Leiden University, The Netherlands

Ruben van Bodegom
rvbodego@liacs.nl

August 20, 2010

Contents

Chapter 1

Introduction

This thesis is about the automated classification of medical reports by a computer program. Our ultimate goal is to design a program which can support medical practitioners in the annotation of their medical reports with a disease code. Currently this annotation of medical reports is done by human experts, which is a time consuming process. Because of natural variation in human judgment it is not uncommon for multiple annotators to assign different codes to the same text. Hospital and insurance company guidelines therefore demand that the annotation process be done with utmost precision, and that reports are not coded unless the code is certain.
Research on automated classification of medical reports has received increasing interest over the last decade. The reduction of the administrative burden upon medical specialists, and thereby the reduction of overhead cost, being an important reason. It is estimated that in 2007 in the US alone $25 billion was spent on the coding of clinical text. Besides this it also makes for a challenging research topic because medical reports are known for their unique sublanguage characteristics, namely verb less sentences, domain specific punctuation semantics and unusual metonomies. These characteristics make automated classifying a very challenging task to perform.
In our research we had two different datasets available. One with medical reports in English which contained all sorts of radiology reports, and one set in Dutch with reports that all came from people who were suspected of acute appendicitis. Both datasets contain several hundreds reports. To classify the reports we make use of some well-known approaches and we have developed several algorithms which incorporate some of our own ideas. Starting with a simple bag-of-words representation we have several different ways to define and calculate the amount of similarity between the reports as well as multiple ways to use this information to come to the conclusion which class the report belongs to. The results are encouraging, but further research will be

necessary to develop them further and see if they work in the general case.
The thesis begins with some background information about Artificial Intelligence, and document classification in particular, as well as the classification used, in Chapter 2. In Chapter 3 we will discuss some related work, in the domain of medical reports, as well as some of the research done in the document classification domain. Chapter 4 will clarify the reason we chose to do this research and the goals we are aiming for. Chapter 5 will describe the problems we faced with the medical reports themselves and with the classification. Chapter 6 provides the solutions we have found to tackle these problems and Chapter 7 goes into some of the implementation details. Chapter 8 shows the results and in Chapter 9 the conclusions and some recommendations for future research can be found.
This thesis is part of the Master in Computer Science at Leiden University, The Netherlands. I would like to thank my supervisor Walter A. Kosters (from LIACS, the Leiden Institute of Advanced Computer Science) for his support as well as Ying L. O (from LUMC, the Leiden University Medical Centre) for providing us with the medical reports and also being available for our questions.

Chapter 2

Background

2.1 Artificial Intelligence

Artificial Intelligence (AI) is defined as: “the study and design of intelligent agents,” [1]. AI attempts not only to understand how we think but also aims to build intelligent entities. The idea of machines that could think for themselves can already be found in the ancient Greek myths. Formal reasoning by philosophers and mathematicians led to the study of *logic*. Alan Turing suggested that his Turing machine could simulate any conceivable act of mathematical deduction, by simply shuffling only the symbols “0” and “1”. Together with the evolution of neurology, information theory and cybernetics, this led to the idea that it would be possible to build an electronic brain. AI has always been fascinating to people, and it is a popular subject for Hollywood movies. Today it plays an important role in the technology industry, and it encompasses a huge variety of subfields.
Although the name Artificial Intelligence was first introduced in 1956 it has been around since 1943. This first period (1943–1956) is characterized by attempts to model artificial neurons. McCulloch, Pitts, Hebb and Minsky were some of the most influential researchers during this period. It was Alan Turing however who first stated a complete vision of AI in his 1950 article “Computing Machinery and Intelligence”[2], which introduced his Turing test, machine learning, genetic algorithms, and reinforcement learning. In 1956 it was John McCarthy who introduced the name Artificial Intelligence. Newell and Simon developed their “Logic Theorist”, which was able to prove many mathematical theorems. Although the computers were primitive and possibilities of programming tools at that time were also very limited, the period from 1956 till 1969 was one in which AI booked quite some success. Being a very new area of research it was not automatically accepted by the

intellectual establishment, and it was during this period that many things not thought possible by the establishment were demonstrated to be possible. Also the first attempts to be competitive in the game of checkers saw the day-light. McCarthy developed the *Lisp* programming language, which became the dominant AI programming language. SHRDLU was one of the programs that lead to great enthusiasm about the possibilities of AI. All these systems showed very promising performance on simple examples. However when faced with more realistic problems they failed miserably. Cuts in funding for AI research from national governments in the 1970s led to the first "AI-winter". In the 1980s, with the introduction of expert systems, AI grew out from a few million dollar industry in 1980 to one of many billions of dollars in 1988. A second AI-winter began when the high expectations failed to be fulfilled, as well that the introduction of the personal computer caused the collapse of the Lisp machine market. The rapid increase of PC power however resulted in renewed interest in the mid 1990s. AI also has focused more on solving subproblems, and has increased its links with other research disciplines.
Our interest mainly focuses on the progress made in the fields of *Natural Language Processing* (NLP), *Information Retrieval* (IR) and *Data Mining* [3, 4, 5]. NLP studies the problems of automated generation and understanding of natural human languages. IR is experiencing a regrowth in interest since the use of Internet searching has become widely popular. Manning et al. [6] give an overview of both traditional IR and web-based search. Data mining focuses on discovering trends and patterns within large sets of numerical data. Each year several conferences are held around the world on the topic of Information Retrieval, and speech and natural language processing.
At present AI is split up into different subdomains. These subdomains are not set in stone and are therefore subject to discussion. Some examples of AI subdomains are: search, logical AI, pattern recognition, common sense knowledge and reasoning, planning, genetic programming, social intelligence and motion and manipulation. Some examples of applications of AI are game playing, of which the game of chess has received most attention when, in May 1997, IBM's Deep Blue supercomputer beat the reigning World Chess Champion, Garry Kasparov, in six matches. Other areas where AI applications play an important role are: robotics, speech recognition, computer vision and heuristic classification.

2.2 Document Classification

Text mining is "The process of finding useful or interesting patterns, models, directions, trends, or rules from unstructured text" [7]. In other words, extracting previously unknown information from textual documents. Text mining can be seen as a subfield of the slightly older and broader field of data mining [5]. Data mining usually deals with structured data from which trends and patterns have to be discovered. Text however is usually fairly unstructured, so the challenge is to identify certain structure within text to allow us to use data mining techniques to be used. Text mining is more than a simple search within a group of textual documents, e.g., the one we typically see in web search, because it actually gives us more information about text documents than we previously had. Document classification (sometimes called text categorization) is one of the subproblems within the text mining domain, which focuses on organizing textual documents into predefined categories or classes. Other typical text mining tasks include document summarization, ranking, text filtering and text visualization.

One of the first references which expresses the need to organize data was as early as Vannevar Bush in 1945, when he proposed the use of his Memex machine to deal with the "growing mountain of research". In 1961 Lauren B. Doyle stated: "natural characterization and organization of information can come from analysis of frequencies and distributions of words in libraries" (where libraries is somewhat loosely used to identify what we now call corpora, i.e., collections of documents). Text mining as a field in computer science therefore might be relatively new, but the dream to be able to extract information from large sets of data has been around for quite some time. The first documented attempts in modern text mining originate from the mid 1980's when people started the very labor intensive process of manual text mining. Since that time more and more textual data has become available, mainly due to the increasing popularity of the Internet and automation using computers in general, which still causes an immense growth of available unorganized textual documents. The introduction of blogs, microblogs such as Twitter and social networking services as Facebook and LinkedIn, as well as the continued increase of use of email, greatly increase the amount of textual data every day. Common estimates show that currently around 80% of the world's data is held in unstructured documents [8]. This includes all business relevant information, mostly stored in reports, email, news- and discussion groups. It doesn't look like the world of information will stop growing or become less complex any time soon!

In the 1980's and 1990's texts were mainly analyzed using NLP techniques [9]. After this initial phase there was general consensus that it involved more than NLP, and Document Classification became a true interdisciplinary part of computer science, involving data mining and machine learning techniques. Document classification consists of several different tasks: simplifying and structuring the input text, deriving patterns within the structured data and finally evaluation and interpretation of the output. There are basically two ways to do this: supervised document classification and unsupervised document classification. Supervised document classification uses some form of external feedback (such as a human expert) to classify a training set. From this some model to map input (text) to output (a class) is generated. Unsupervised document classification uses no external information about the classification of the documents, this is often referred to as document clustering, as there are no predetermined categories but one merely tries to cluster documents together that are similar.

The first step in document clustering is usually preprocessing, which is meant to reduce the complexity of the documents. It focuses on *noise removal* as well as finding a suitable representation for textual documents. Then the documents have to be transformed from their full text version to a document vector. One of the major challenges in document classification is the extreme high dimensionality of textual data. The number of document features not seldom exceeds the number of training documents, mainly caused by the complex semantic structure of text, such as word order and punctuation marks, as well as the occurrence of synonymy (several words with the same meaning) and polysemy (one word having multiple meanings). There are several techniques to reduce the dimensionality of the documents somewhat: the removal of “noise”: stop words (words such as “the”, “a”, “and”, etc. occur so frequent in any text that they have no meaning when looking at text similarity) as well as the stemming of words (conversion of words into their basic form: connection, connecting and connects are all mapped to connect) are commonly used.

Another step in the simplification process is *feature selection*. The “term frequency-inverse document frequency” (TF-IDF) approach is one of the most popular methods used to give a weight to each word according to its uniqueness, in other words the relative frequency of the occurrence of a word in a report in respect to the entire corpus. Other examples of feature selection methods are: information gain, Chi-square, expected cross entropy, odds ratio, mutual information and Gini index [10]. Some of these will be explained in Section 3.2, when looking over some related work. One of the oldest techniques is to simply treat a text as a *bag-of-words*, simply counting the occurrence of separate words. More advanced techniques look at simple linguistics as well as varied word forms, such as abbreviations, plurals and conjugations as well as multi-word terms, called n-grams. Complete use of the semantic relations within text seems impossible, due to the high complexity nature of text. Finding the right representation remains one of the biggest challenges in text mining, and often the simplest methods still prove to give good results.

After the text documents have been reduced to a word vector, the second stage of document clustering takes place. This is the *machine learning* phase. As described above this can be done using supervised learning or through unsupervised learning. Both involve the use of machine learning algorithms, which is an area that receives a lot of interest in the research community. We focus on the use of supervised learning algorithms, where pre-defined category labels exist. Some different approaches include: (artificial) neural networks, genetic programming, decision tree learning, support vector machines, k-Nearest Neighbor and fuzzy correlation are examples of some of the more commonly used techniques. Some of these techniques will be described in Section 3.2, when we are looking at related work. Our approach uses the *k-Nearest Neighbor* algorithm. This compares the degree of similarity from a text to k (where k is a pre-determined natural number) reports from the training data. The idea is that the properties of a report x are likely to be similar to the properties of a report in the neighborhood of x. This sounds very straightforward and trivial until we think a bit more about the term "neighborhood". If we look at a neighborhood that is too small we might not find any other data points, if our neighborhood is too big it can contain a very large subset of data points. By using a fixed number k we solve this problem: where data is dense the neighborhood will be small and where data is sparse the neighborhood will be large, see Figure 2.1.

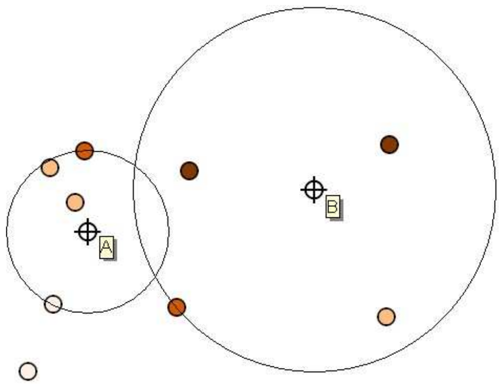

Figure 2.1: Example of the k-Nearest Neighbor algorithm, where $k = 4$.

To be able to talk about neighborhoods we also need to define a way to measure similarity, in other words: What is meant by *distance*? Defining distance or similarity is the challenge of any implementation of the k-Nearest

Neighbor (KNN) algorithm. The training phase of the KNN algorithm consist only of storing the feature vectors and their corresponding categories of the training set. In the classification phase the distances from the new documents feature vector to all stored vectors are computed and k closest samples are selected. The category of the new document is derived from the categories of those selected documents, usually by simply selecting the most common category among the k nearest neighbors. This method is quite easy to implement, requires little tuning and performs quite well [1]. Finding the optimal value of k can be challenging, and classification time can get rather long, especially when we are dealing with large datasets. More preprocessing would then be necessary to reduce the calculation time by either preventing the calculation of distances to all points in the dataset or by further reducing the dimensionality of the data [11].
Applications of document classification (DC) are:

- Email filtering
  Email can be filtered to prevent the inbox from being overflooded by spam, but it can also be used to categorize incoming mail into folders, based on the words in the subject field or the group of recipients.

- News monitoring
  Companies that rely heavily on getting the latest news and information, e.g., those in international economics or stock exchange, currently have employees which scan through newspapers and other sources. This process can be automated by DC.

- Mail routing
  Large enterprises use something called work-flow management to allow incoming documents, both digital as well as in paper form, to circulate between relevant employees. Getting documents in the right work flow is a major task, which can be automated by using DC.

- Automated classification of scientific articles
  With the ever growing amount of scientific reports and articles, the interest in finding relevant documents also grows. Keywords and information from an abstract or introduction can be used to categorize or cluster relevant articles together for better retrieval than through an ordinary web search.

2.3 International Classification of Diseases

The *International Classification of Diseases* (ICD) is the international standard for all general epidemiological, many health management purposes and clinical use. Within this international standard there is some room for national health organizations to create their own entries. The *World Health Organization* (WHO) [12] has been responsible for the ICD since its creation in 1948 when the sixth revision was published. The tenth revision, *ICD-10*, has been in use since 1994. The classification itself dates back to the 1850's when it was known as the International List of Causes of Death, which became a standard in 1893. It is used to classify diseases and other health problems recorded on many types of health and vital records including death certificates and health records. It also allows the WHO to generate mortality and morbidity statistics for its member states.

The ICD-10 classification consists of 22 chapters. Each of these chapters contains a subset of diseases. An overview of the different chapters in the classification can be found in Appendix A. As can be seen from this table some diseases are classified by the type of disease, while other are classified by the affected organ(s) or functional system. Each ICD-10 code consists of a letter, followed by two numbers and optional one or two decimal numbers. The hierarchical structure of ICD-10 can be seen in the following example concerning Acute appendicitis. We can distinguish four levels of specificity. Acute appendicitis on the highest level is a disease of the digestive system, thus being in chapter XI (K00-K93). On the second level we have all diseases of the appendix in the K35-K38 range. On the third level K35 deals with acute appendicitis, with the subclassification on the fourth level found in Table 2.1.

The size of each level within this classification as well as the order in the ICD-10 classification are arbitrary, thus resulting in a so-called *nominal or categorical* level of measurement as defined by Stevens in a 1946 Science article [13]. This means that there is no intrinsic ordering of categories, i.e., the order is not based on being better or worse, or any other quantitative measurement.

K35	Acute appendicitis
K35.0	Acute appendicitis with generalized peritonitis
	Appendicitis (acute) with:
	perforation
	peritonitis (generalized)(localized) following rupture or perforation
	rupture
K35.1	Acute appendicitis with peritoneal abscess
	Abscess of appendix
K35.9	Acute appendicitis, unspecified
	Acute appendicitis with peritonitis, localized or NOS
	Acute appendicitis without:
	generalized peritonitis
	perforation
	peritoneal abscess
	rupture

Table 2.1: Example of the ICD-10 classification of acute appendicitis.

Chapter 3

Related work

A lot of research is done on document classification. We are going to discuss some relevant research done, where we focus on classification of medical reports on one hand, and on the other hand we look at some of the different techniques which are popular in the document classification domain.

3.1 Medical reports

Due to the nature of medical reports it is believed by some [14] that proper classification is highly dependent on the use of domain knowledge. The cost involved in adding this domain knowledge to classification algorithms, however, sometimes exceeds the cost of manual classification. Therefore there will always be a trade off between having a self-learning algorithm and the input of knowledge from human experts.
One branch of research, referred to as Medical Informatics, focuses on automatically extracting information from medical reports to allow the creation of a more structured form to store the clinical data. This makes them suitable for use by automated decision support systems. An example of this would be to extract certain clinical conditions from the text and to store these in separate fields [15]. To do this an extra preparation step is introduced before the actual document classification process begins.
Another approach is to use domain experts in the feature selection process to select specific attributes or features, or even to combine certain features together. Although this induces extra cost to the feature selection process it also leads to a reduced number of variables used and thereby the complexity.
Most of the research done is on datasets which are highly pruned. They contain several hundreds of medical reports, but the reports belong to a very limited number of classes. This allows for very good results and therefore isn't

really useful for a comparison with the situation we have. Recently there has been a shared task on a larger dataset [16] and, although the authors also selected a subset of only 45 ICD codes, it does show some promising results.

3.2 Document classification

When looking at recent research done in the general document classification domain, it is obvious that all researchers focus on one of the different tasks within the classification process. This has become necessary because the domain has already developed to an extent where research on the entire process simply becomes to comprising. To give an overview of the available techniques we will therefore look at each of the different tasks necessary to get from a textual document to a correct classification. First we will look at ways to translate the textual data into a feature vector. Then we will look at ways to compare these document vectors and examine different clustering techniques. And finally we will discuss some alternatives to decide, from this selection, what the category of our new document should be.

3.2.1 Document representation

The first step which allows us to compare our medical reports is to find a suitable representation for our textual documents. We need to find some structure which allows our algorithms to calculate how similar our medical reports are. We want to translate our raw text medical reports into a so-called feature vectors. The vector stores which features are existent within a report. There are two types of feature selection methods used in machine learning: wrappers and filters. *Wrappers* use the results of their own subset selection to train a classifier. The subset that receives the best results is then chosen as the final set. As a result wrappers have to train a classifier for each feature subset and are therefore very time consuming, especially when the number of features is high, which is generally the case in text classification. Wrappers are thus considered to be unsuitable for text classification. *Filters* perform their feature selection independent of the learning algorithm. A common approach is the Term frequency/inverse document frequency (TF-IDF), where each word is weighted on how unique it is. Some of the more recent work focuses on: Chi-square, information gain, ant colony optimization, Gini index, Poisson distribution, and expected cross entropy, are only a few examples [10, 17].

3.2.2 Classifier construction

The second phase consists of machine learning. In text classification this is usually done by means of supervised learning, i.e., pre-classified documents are used to generate a function which maps unclassified documents to the desired class. Rocchio's algorithm, k-Nearest Neighbor, decision trees, Naive Bayes, decision rule, regression methods, neural networks, support vector machines, and fuzzy correlation, are some examples of the popular methods used in current day research. Increasing in popularity are so-called hybrid techniques, which combine two of the before mentioned techniques, one for the vectorization and another for the classification. Support vector machines, Naive Bayes and k-Nearest Neighbor appear to receive the most attention in this context [10].

3.2.3 Classifier evaluation

Evaluation of document classifiers is typically done experimentally, rather than analytically. The reason is that, if we wanted to evaluate it analytically, we would need a formal definition of what correctness and completeness are in this context. The idea of text classification, namely assigning a document to the membership of a category, is nonformalizable due to its subjective character. Therefore evaluation of a classifier can only be done by experimental means. Evaluation of the results is often done by calculating precision and recall. Both of these are calculated on a per class basis. When borrowing terms from logic, precision can be viewed as the degree of soundness, while recall is the degree of completeness. Recall and precision always have to be viewed together to give a good view of the performance of a classifier, and are therefore often combined in the F-score, which is the harmonic mean of the precision and recall.

Chapter 4

Project motivation and goal

Automated document classification is an area of research that receives a lot of interest nowadays, mainly due to the increasing amount of digital data. More sophisticated methods are available then ever before and new ones are being developed continuously. Automated classification of medical reports raises some ethical questions. People generally don't like the idea that computers, or other devices that they can't control or understand the working of, have an influence in their medical diagnosis or treatment. For the time being it therefore looks as if the role of the computer is limited to that of an instrument for the medical personnel, a tool that can help them take certain decisions. Once these kind of tools have proven their value maybe the scepticism will make place for enthusiasm.

Our research has two main goals, both with their respective subgoals. The first goal is to be able to classify the medical reports such that we can utilize the program to predict the correct ICD-10 code of a new medical report. We, as computer scientists, have no knowledge of the medical jargon. Of course we could have developed this, but we instead decided that we wanted to write a program which had no prior knowledge of the data it was going to receive, thereby making it as versatile as possible. The only expert knowledge that we used was the classification which was in the set of medical reports which we had.

Our second goal is to make use of well-known, existing methods and techniques but mainly to be creative and try to find new ways to create a proper classification. Thinking out-of-the-box and experimenting with different ideas which are not part of the mainstream thoughts is done explicitly. We have been able to come up with several new ideas for comparing the similarity between reports. Because we want to introduce our own ideas to the program, we needed to program it ourselves to have full control of what is going on inside.

Because we were aiming for two different goals we had to be aware not to sacrifice one for the other. Getting the best possible classification and having the program be relatively simple and modifiable has been quite a challenge. During our research on a dataset of Reuters newsreports, we achieved an accuracy of up to 77% when running it on 1,000 reports and up to 85% when running it on 6,000 reports [18]. Because both medical datasets we have only contain several hundreds reports we are not expecting to get results that high.

Chapter 5

Problem definition

5.1 Medical reports

Before we are going to look at the challenges we face when analyzing our medical reports, we want to get a better idea of what these reports look like. We have two different datasets:

- General radiology reports (in English)
  A collection of 380 reports from a University hospital in the United Kingdom. All reports originate from the radiology department, but they deal with a wide range of diagnoses. Some reports have been diagnosed as belonging to several different ICD-10 codes. In total there are 627 diagnoses. A report therefore has an average number of diagnoses of 1.65. In reality most reports have either one or two related ICD-10 codes, while only few have more. See Table 5.1 for more details.

No. of ICD 10 codes	No. of reports.
1	211
2	125
3	24
4	11
5	4
6	3
7	1

Table 5.1: Distribution of the number of ICD-10 codes per report.

This implies that we are looking at a so-called *multilabel* case, where the

mapping of one document to multiple categories is possible. This automatically implies that the categories are overlapping. In the Document Classification domain it is common to assume that the multilabel case is a special case of the single-label or *binary document classification*. For this to be true the assumption that all classes are independent of each other has to hold. For medical reports however this isn't necessary the case. In fact it might be plausible that some pairs of diagnoses are quite common due to the similarity of the disease or of the area being looked at by the radiologist. Having to deal with multilabel categorization complicates the classification.
Below is an example report, labeled as D18.0 and M51.4:

```
1234567 01/01/2001 CT SPINE THORACO LUMBAR \par
   1234567 01/01/2001 CT SPINE LUMBAR \par
   Indications: Lesion in L4 on MRI ?nature.
   Possible haemangioma in T11 \par Technique:
   Spiral scans were obtained from T10-T12 and from
   L3-S1 \par Findings: \par Comparison was made
   with the MRI from Jan 2001 \par There is
   degenerative change in the lower thoracic and
   lumbar spine. \par No destructive bony lesion in
   the lower thoracic spine. \par There is a defect
   in the superior end-plate on the left side of the
    L4 vertebral body with surrounding sclerosis,
   corresponding to the region of abnormality
   previously described on MRI. The appearances
   would be consistent with an end-plate infarction
   with formation of a large Scmorl's node. It is
   reassuring that the overall dimensions of the
   lesion have not increased significantly since the
    last scan although a defect has now developed. \
   par Conclusion \par Features consistent with
   evolving end-plate infarction. \par Findings
   discussed with Drs Abcdef and Ghijkl \par
```

A `\par` in the medical report denotes the beginning of a new paragraph. When reading several reports like this one it becomes clear that a lot of medical terminology is used, and that those findings that are normal or expected are only briefly noted, while the explanation on abnormal findings is much more elaborate.
Another issue with these reports is that we have only 380 labeled reports, and that there are several thousands possible ICD-10 codes. In

our training set alone we are already looking at 278 unique ICD-10 codes. When we examine the most common diagnoses we see that only seven of them occur in ten or more reports. Using our approach we will probably need a more densely populated set of reports.
Another way to look at the classification is to use the area which is affected by the disease, this could be an organ or a certain body part. The ICD-10 classification also classifies each disease using a ICD-10 region. This does not solve the multilabel case issues we talked about before, but it does reduce the number of classes. When we look at the distribution of the reports over the different regions we see that we have a much better report to class ratio than we have with the ICD-10 codes. We are also facing 67 unique classes instead of the 278 unique ICD-10 codes, and we have seven classes now for which we have more than twenty reports in our training set. Because of this we are also going to classify our reports on the different regions involved in the research instead of just on the ICD-10 codes. For more details and an overview of the similarities and differences between the two approaches the reader is referred to Table 5.2 at the end of this section.
Because of the characteristics of the dataset it is going to be referred to as the *heterogeneous* dataset in the rest of this thesis.

- Reports on suspected appendicitis (in Dutch)
  The second dataset contains medical reports written in Dutch. It contains 442 report on people with suspected appendicitis or related problems. In contrast with the English reports we are dealing with a *singlelabel* case here, i.e., each report belongs to only one class. Therefore there are also 442 diagnoses. Because all these reports are about a very specific part of the body there are also much less different ICD-10 codes used, only 29. The reports should be much more similar when compared to the English reports. This can turn out to be an advantage as well as a disadvantage. It can be an advantage because the density of reports will be much higher, thus allowing us to better compare a new report to already existing reports. It can also be a disadvantage because the classes could be so overlapping that we will not be able to distinguish between them. Below is a report that is classified as K35.9: "Acute appendicitis, unspecified".

```
Relevante klinische gegevens Buikpijn en koorts.
   Verdenking appendicitis, leukocytose, druk en
   loslaatpijn McBurney. Verslag Er is geen oud
   onderzoek ter vergelijking. In de
   rechteronderbuik appendix welke proximaal een
```

```
diameter heeft van 6/7mm. Duidelijke gelaagdheid
van de wand. Mediaal van de epigastrische vaten
veel vet infiltratie waarin het distale deel van
de appendix gelegen is met een diameter van 1,4cm
, hierin bovendien een hyperechogene structuur
met slagschaduw, passend bij appendicoliet. De
appendix is niet comprimeerbaar. Enig dbris in de
 appendix. Tevens veel vrij vocht in de omgeving,
 dit kan indirect teken zijn van perforatie.
Reactieve enigszins verdikte wand van het
terminale ilium. Normaal aspect van nieren
beiderzijds. Spoortje vocht in Morrison pouch.
Normaal aspect van milt, lever en galblaas.
Normaal aspect van het pancreas. Conclusie
ECHOgrafisch beeld van appendicitis acuta,
mogelijk perforatie. Uitslag besproken met
piepernummer 00000. Einde verslag.
```

When comparing reports from this dataset to the reports in the heterogeneous dataset it is clear that their structure is much more loosely defined. Because these reports are quite similar due to the expected disease, this dataset will be referred to as the homogeneous dataset.

5.2 Feature Selection

Now that we have an idea of what our reports look like we have to think about the challenges we will face when interpreting these reports. For our program to be able to compare the reports we have to find a suitable representation for the textual documents first. We are going to look at certain features (or properties) of the documents, and store them in a so-called *feature vector*. We first have to decide which features in the reports we are going to consider, and then for each report we are going to store how frequent each feature occurs.

Several design decisions have to be made for the selection of the features we want to consider. Firstly, how do we look at the reports? Do we consider sentences as a whole, do we look at combinations of words or do we ignore the order of words and only count their number of occurrences? Secondly there is the matter of disavowal words (no, not, non, none, etcetera) which refer to another word. It is hard to determine which word they refer to, it could be the word that immediately follows, but it could also be somewhere

Dataset	Heterogeneous		Homogeneous
Classification done on:	ICD-10 code	ICD-10 region	ICD-10 code
Number of reports	380	380	442
Avg. report length	94 words	94 words	93 words
Total number of diagnoses	627	661	442
Avg. number of diagnoses/report	1.65	1.74	1
Number of unique diagnoses used	278	67	29
Top 10 diagnoses with frequency:			
1.	I26 (38)	lung (156)	NSAP (90)
2.	J98.1 (19)	brain (84)	K35.9 (81)
3.	J90 (15)	kidney (59)	K35.0 (65)
4.	R65 (15)	abdomen (32)	K35.1 (56)
5.	K80.2 (11)	intestines (28)	K52.9 (38)
6.	N28.1 (10)	lymph (25)	I88.0 (19)
7.	I63.9 (10)	liver (24)	K57 (17)
8.	N20.0 (9)	pancreas (17)	K59.0 (9)
9.	R59.0 (8)	gallbladder (15)	N83 (7)
10.	J47 (8)	thorax (10)	N94 (7)

Table 5.2: Some statistics about our two different datasets, and also on the two different approaches within the Heterogeneous set.

else in the sentence, or even outside of the sentence. Lastly there is the issue of the importance of words. Some words (the, it, a, an, to, etc.) are very common and occur in almost every article. Therefore they do not help when comparing reports. So maybe we should ignore them?
The first step is critical for the behavior of the rest of the program. If we choose to ignore the order in which words occur and only count their existence we use the so-called *bag-of-words* model. The feature vector simply consists of a list of word and their frequency. If we look at words (or even characters) and also the words surrounding them we use the so-called *n-grams*, which are commonly used in statistical natural language processing. An n-gram looks at a (short) sequence of words. If $n = 3$ than each feature would consist of a series of three words. The processing of sentences as a whole still creates many practical problems.

5.3 The different ways to measure distance

When we have decided what our feature vector is going to look like, we can start to think about ways to compare these vectors. The problem here is the high dimensionality of the feature vector itself, it usually contains tens to hundreds of features to be compared. If we want to be able to discover the reports in our database that are most like a new report we have to find a way to describe and quantify similarity between their feature vectors. We are going to define this as the distance between two feature vectors: the smaller the distance between two vectors, the more the two reports are alike, and therefore the bigger the chance they deal with the same disease. So what is the definition of the word distance in terms of natural language reports, i.e., how do we find out which reports are more similar to each other than others? The answer to this question is usually tough enough to answer for a human expert and more often than not if you ask several experts they come up with different answers. So finding a definition for this distance will be a tough challenge for our program. After this is done we also have to find ways to calculate the distance between our natural language reports. Because of the complexity of this issue we might look at several different algorithms for calculating distance.

5.4 Determining the ICD-10 code

Now that we have a sorted list of the distances between a new report and all existing reports in the database, we need to determine which ICD-10 code belongs to the new report. To do this we first have to determine how many, and which, reports in the list we are going to consider. We are using several different strategies to make this selection, both fixed and dynamic. After that we can look at the different codes of the selected reports, and decide how we are going to weigh each of them in order to find the ICD-10 code of our new report. In this last step we also use several different techniques to determine the most probable ICD-10 code of the new report.

Chapter 6

Solution

6.1 Feature Selection

Before considering automatic classification of our reports we had to make a translation from text to a so-called *feature vector*. We used several different approaches:

Classic Bag-of-Words

In the classic bag-of-words only the frequency of a word within a report is considered. The word order, position, or the semantic relation between words is ignored. Although clearly an oversimplification, research shows that the results of this method are still quite good. In Figure 6.1 is an example of a word X that occurs in 8 positions in our report. This example will be used for all our other feature selection methods in this report as well.

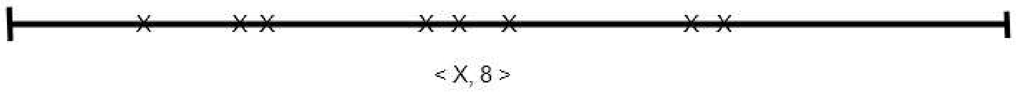

Figure 6.1: Example of the classic bag-of-words model.

Multiple Bags-of-Words

Our medical reports each consist of three parts. First there is the clinical information, which contains the medical background of the patient as well as the symptoms for which research takes place. Second there is

a section with the findings in which the doctor gives a description of what (s)he sees. The report ends with a conclusion in which the doctor draws conclusions from his/her findings.
In this approach we treat each of these three sections separately, and therefore use three bags for each report. Each possible combination of one, two or three of these bags has been considered. For example we have only compared the clinical information and the conclusion, or just the conclusions, of two reports. In the example in Figure 6.2 the occurrences of word X for each section of the report are stored, so 2 for the first section, 5 for the middle section and once in the last section.

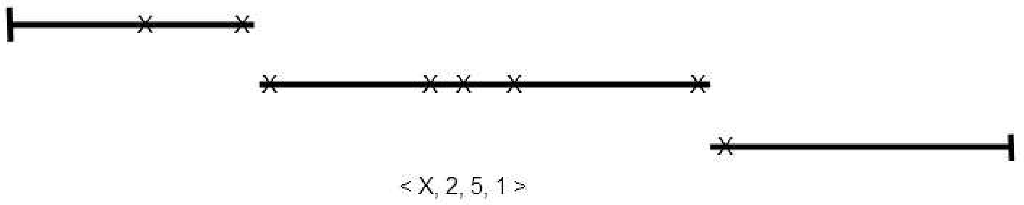

Figure 6.2: Example of the multiple bag-of-words model.

Bag-of-Intervals

This approach has gone beyond only counting the number of occurrences of a word. It looks at the words in the nearby surrounding and if the same word is within a certain distance it is treated as one occurrence. The distance which determines wether two occrunces are considered as one is called the *worddistance-parameter*. In our example in Figure 6.3 the frequency of X is 4.

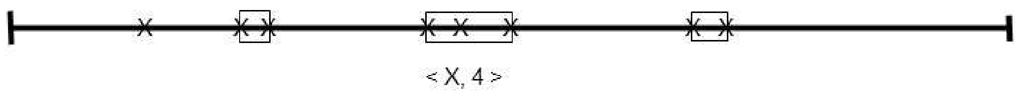

Figure 6.3: Example of the bag-of-intervals model.

Bag-of-Intervals with positions

The same approach as above was used for the bag-of-intervals with positions, however it also compares the positions within each report where a word occurs. To be able to compare reports of different lengths it was necessary to normalize each report to a fixed length of 100, which is close to the average length of our reports. After this normalization the positions of the intervals were compared. An outer margin around these intervals was allowed which was used when looking for corresponding intervals in another report. This margin is called the *margin-parameter*. In our example in Figure 6.4 it can be seen that we now store the word in combination with the begin- and end-position of the interval. So <X, 12, 16, 21, 28, ... > means that the word X has an interval from position 12 to position 16 and also from position 21 to 28.

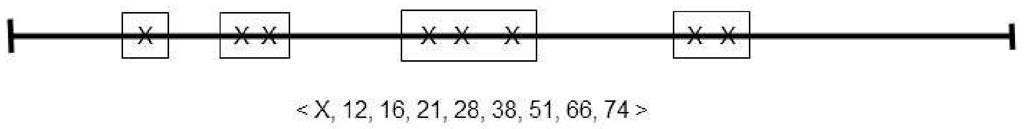

Figure 6.4: Example of the bag-of-intervals with positions model.

The margin-parameter has been set to roughly half the length of the worddistance-parameter. The reason for this becomes clear when thinking of a simple example with two words that are just within the maximum worddistance of each other, they both virtually contribute to half of this distance. So the maximum margin-parameter (which is on the outside) should be the same as this individual contribution: half the length of the worddistance. Figure 6.5 shows two words that are just within the worddistance of each other and shows the margin around the interval as well.
The algorithm used to define these intervals from the occurrences of a word can be found in Figure 6.6.

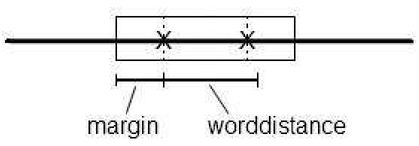

Figure 6.5: The margin- and worddistance-parameter.

```
input
    $position$: positions of a given word $W$
    $nr$: number of occurrences of $W$
    $Wmargin$: the worddistance margin
output
    $boundaryL$, $boundaryR$: left and right interval boundaries
    $interval$: number of intervals

begin
    $interval \leftarrow 1$
    for $current \leftarrow 1$ to $nr$ do
        if $current = 1$ then
            $boundaryL[interval] \leftarrow position[current]$
        else if $position[current] - position[current - 1] > Wmargin$ then
            $boundaryR[interval] \leftarrow position[current - 1]$
            $interval$++
            $boundaryL[interval] \leftarrow position[current]$
    $boundaryR[interval] \leftarrow position[current]$
end
```

Figure 6.6: Algorithm — Creates the intervals from a list containing the positions of a word.

6.2 The different ways to measure distance

In each of the first three ways to represent our report: classic-bag-of-words, multiple bags-of-words and bag-of-intervals, the feature vectors contain <word, number_of_occurrences> pairs. The distance between pairs of articles was calculated in four different ways. In other words: four different definitions of distance have been used. Following is a description of these definitions and their characteristics. In the definitions below, A and B are the feature vectors of the two reports being compared, which can also be seen as sets.

Hamming distance

This is the easiest way to measure the distance. It is the sum of the number of unique words that occur in only one of the two reports. It treats the bag-of-words model as a simple set-of-words, i.e. we only look at it in a binary way, the word either exists or it doesn't. We are not interested in the number of occurrences.

$$D_{Hamming}(A,B) = |\,A \setminus B\,| \;+\; |\,B \setminus A\,|$$

Here $A \setminus B$ are the features (words) that are in A, but not in B.

The Hamming distance is a metric. Indeed with $D = D_{Hamming}$:

$$\begin{aligned} &D(A,B) \geq 0\ ,\ D(A,B) = 0 \iff A \;=\; B && \text{(positive definite)}\\ &D(A,B) = D(B,A) && \text{(symmetry)}\\ &D(A,B) \leq D(A,C) + D(C,B) && \text{(triangle inequality)} \end{aligned}$$

Jaccard or Weighed Hamming distance

The "normal" Hamming distance, described above, only counts the number of unique words that appear only in report A or only in report B. It does not take into account the number of unique words that the two reports have in common. The Jaccard distance does take this into account. The Jaccard distance divides the number found in the Hamming distance by the number of unique words that occur in either of the two reports. By doing this we normalized for the size of the report:

$$D_{Jaccard}(A,B) = \frac{|\,A \setminus B\,| \;+\; |\,B \setminus A\,|}{|A \cup B|}$$

The Jaccard distance is also a metric. To clarify the difference between the Hamming distance and the Jaccard distance we take a look at the following two examples.
First we assume that there is a report A that contains 50 unique words. Report B also contains 50 unique words. Report A and B have 10 unique words in common. The Hamming distance for this example is

$40 + 40 = 80$. The Jaccard distance is $\frac{80}{90} \approx 0.889$.
Now we take a look at another example. Assume report C and D each contain 270 unique words of which they share 230 unique words. Intuitively one would say that the distance between C and D (sharing 230 out of 270 unique words, which is over 85%) is less than the distance between A and B (sharing 10 out of 50 unique words, which equals 20%). The Hamming distance between C and D however is the same as that between A and B: $40 + 40 = 80$. The Jaccard distance is $\frac{80}{310} \approx 0.258$. Because the Jaccard distance takes the number of shared words into account it is expected that this will give better results for the distance between reports.

Multiset distances

Both the Hamming distance and the Jaccard distance use the so-called binary representation for comparing the reports. This means they only look at the words on an existing or non-existing basis. They don't take the number of occurrences of words into account. We take a look at several examples to make this clear.
Example 1: a word occurs once in report A and once in report B. This is obviously a perfect match.
Example 2: a word occurs five times in report A and also five times in report B. This is once again a perfect match, or is it perhaps even better than Example 1?
It becomes more interesting when the number of words is not the same. We look at another two examples.
Example 3: a word occurs once in report A and ten times in report B.
Example 4: a word occurs five times in report A and ten times in report B.
For the Jaccard distance all of these four examples are the same because the word occurs in both reports. We however ask ourselves the question if this is correct, isn't Example 4 better than Example 3? And what about Example 2 and Example 1?
To allow us to distinguish between those examples we used the so-called multiset distance, [19]. A *multiset* is a set in which repeated elements are considered. Therefore with each unique word (a member) we also stored the number of occurrences in each report; this number of occurrences is called the *multiplicity* of the member. The multiset is sometimes also called weighted set. In the normal sets we are only interested in the existence of members, not their multiplicity. We define the multiset distance as follows:

$$D_{multiset}(A, B) = \frac{\sum_{i=1}^{n} |f(a_i) - f(b_i)|}{|A \cup B|}$$

The multiset distance takes the sum over all unique words occurring in at least one of the reports A and B. This sum is the absolute difference of the values of function f in a_i and b_i, where a_i and b_i are the number of occurrences of the i-th word from our list in report A and B, respectively.
We use two different functions for f within this multiset distance, thereby creating two different ways to measure distance:

$$f_1(x) = \frac{x}{x+1}$$
$$f_2(x) = 1 - 2^{-x}$$

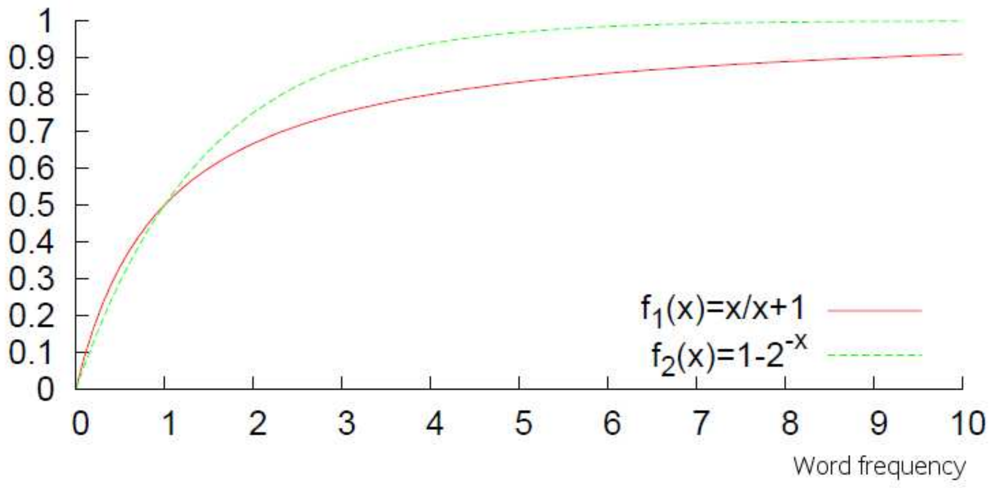

Figure 6.7: The graphs of the multiset functions f_1 and f_2.

For both choices of f the multiset distance is a metric [20]. We will refer to the distance using f_1 as the multiset1 distance and to that using f_2 as the multiset2 distance. Both functions have a range of zero to one, where f_1 and f_2 both approach the limit of 1 as the word frequency grows. But f_2 does this a lot faster than f_1 does, see Figure 6.7. So in f_1 the distance penalty is bigger, especially in the lower frequency region. Note that when we would be using sets here instead of multisets we would get the Jaccard distance.

Interval relation

For the similarity between our *bag-of-intervals with positions* a slightly different approach had to be used. In this case we didn't just have a feature vector A which contained the words and the number of their occurrences but instead, for each word w in A we had a list A_w with the word and the left and right boundaries of each of the intervals. Because of this the comparison between two reports became slightly different as well, it was no longer sufficient to only look at numbers, but it was also necessary to consider the relative location of the occurrences of word-intervals. Each word-interval that didn't have an overlapping interval in the other report adds some value to the relation, where overlapping intervals do not contribute to the relation. However the more overlapping intervals the two reports have the smaller the relation penalty is for a non-overlapping interval, similar to the Jaccard and multiset distances:

$$\begin{aligned}
\text{interval}(A_w) &= \text{the collection of intervals in } A_w \\
\xi(A_w, B_w) &= \{u \in \text{interval}(A_w) \mid \nexists v \in \text{interval}(B_w) : u \cap v \neq \emptyset\} \\
R_{interval}(A_w, B_w) &= \frac{|\,\xi(A_w, B_w)| \;+\; |\xi(B_w, A_w)\,|}{|\text{interval}(A_w)| \;+\; |\text{interval}(B_w)|} \\
R_{interval}(A, B) &= \sum_w R_{interval}(A_w, B_w)
\end{aligned}$$

In other words: the interval relation is the number of intervals which are unique to A plus the number of intervals which are unique to B over the total number of intervals in A and B.
Here $R_{interval}(A, B) = 0$ can hold when $A \neq B$, and also the triangle inequality does not hold in general. An example of this relation measure can bee seen in Figure 6.8.
The algorithm used to calculate the interval relation between two reports is given in Figure 6.9.

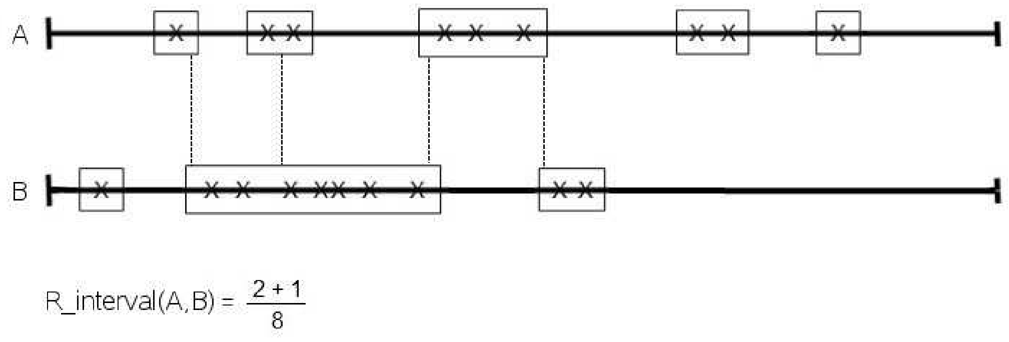

Figure 6.8: Example of the interval relation between two reports using the bag-of-intervals with positions method.

Overlap relations

The interval relation only counts the number of intervals. To be able to do something with the length of the intervals and the amount of overlap, we introduced the *overlap relations.* Two different approaches were used. Overlap relation 1 is defined by the sum of the length of the intervals that are unique to report A and B over the sum of the length of all intervals. Overlap relation 2 is more precise, in the fact that it doesn't only measure the unique intervals, but it also adds parts of intervals that don't have an overlap in the other report, while overlap relation 1 only looks at intervals that have no overlap at all.

$$\begin{aligned} \text{length}(Z_w) &= \text{ sum } \{ \ |z| \ , \ z \in \text{interval}(Z_w) \ \} \\ R_{overlap}(A_w, B_w) &= \frac{\text{length}(\xi(A_w, B_w)) + \text{length}(\xi(B_w, A_w))}{\text{length}(A_w) + \text{length}(B_w)} \\ R_{overlap}(A, B) &= \sum_w R_{overlap}(A_w, B_w) \end{aligned}$$

This is the same as in the situation of the interval relation, only here we look at the length of the intervals instead of the number of intervals

An example of the two overlap relation measures can be found in Figure 6.10 and Figure 6.11, the relation is calculated using the length of the orange parts over the length of all parts.

```
input
    $A.boundaryL, A.boundaryR$: left and right interval boundaries of report A
    $A.interval$: number of intervals of report A
    $B.boundaryL, B.boundaryR$: left and right interval boundaries of report B
    $B.interval$: number of intervals of report B
output
    $RintervalAB$: interval relation between report $A$ and $B$
    $overlap$: number of overlapping intervals between $A$ and $B$
    $nonoverlap$: number of non-overlapping intervals between $A$ and $B$
begin
    $matchfound \leftarrow$ false
    $inA \leftarrow$ false
    $inB \leftarrow$ false
    $A.current \leftarrow 1$
    $B.current \leftarrow 1$
    for $i \leftarrow 1$ to 100 do
        if $A.boundaryR[A.current] = i$ then
            $inA \leftarrow$ false
            if $matchfound$ and not $inB$ then
                $overlap$++
                $matchfound \leftarrow$ false
            else if not $matchfound$ then
                $nonoverlap$++
            $A.current$++
        if $B.boundaryR[B.current] = i$ then
            $inB \leftarrow$ false
            if $matchfound$ and not $inA$ then
                $overlap$++
                $matchfound \leftarrow$ false
            else if not $matchfound$ then
                $nonoverlap$++
            $B.current$++
        if $A.boundaryL = i$ then
            $inA \leftarrow$ true
            if $inB$ then
                $matchfound \leftarrow$ true
        if $B.boundaryL = i$ then
            $inB \leftarrow$ true
            if $inA$ then
                $matchfound \leftarrow$ true
    $RintevalAB \leftarrow overlap \ / \ (overlap + nonoverlap)$
end
```

Figure 6.9: Algorithm — Calculates the interval relation between two reports.

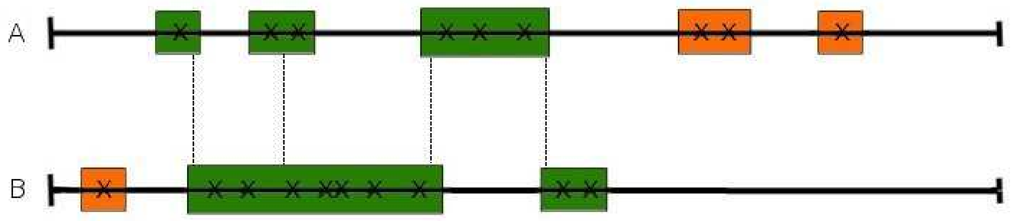

Figure 6.10: Example of the overlap1 relation on the bag-of-intervals with positions method.

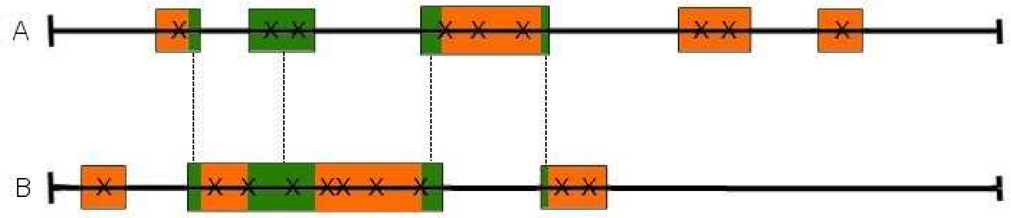

Figure 6.11: Example of overlap2 relation on the bag-of-intervals with positions method.

6.3 Determining the ICD-10 code

Each of the distance functions from Section 6.2 results in a sorted list of the distances between the new report and the reports already in the database. To determine the most probable ICD-10 code we once again use several methods, as was announced in Section 5.4.
The first step was to decide which reports from our list needs to be considered to find the ICD-10 code of the new report. The method used to select the appropriate reports is the well-known k-*Nearest Neighbor* algorithm [1], See also Section2.2. To determine how many reports are going to be consider, i.e., k, we used both a static and a dynamic selection method: “fixed quantity” and “distance boundary” respectively.
In the fixed quantity method the number of reports is fixed and therefore always known beforehand. It simply selects the k closest reports (where k is a pre-determined number). To find the optimal number k, it is necessary

to do calculations with multiple different values of k. See Figure 2.1 for an example.
The other method, distance boundary, selects all reports that are within a certain preset distance of the new report. It results in an unknown number of selected reports in our list (potentially the list can be empty or contain all reports).
The usage of each of these two selection criteria resulted in a shortlist of reports that are considered important to decide what the ICD-10 code belonging to our new report is. This shortlist was used to make an educated guess on the code of the new report. Several different wighing schemes were used to determine which code is most likely to belong to the new report. The first, and most basic, method is to just pick the code that occurs most frequently in our shortlist. Alternatively the codes were weighed: giving the codes of the closest report a higher score than those further away. This was done using a fixed score, where the closest reports ICD-10 code receives a score of k, the second closest a score of $k - 1$, etc. The third alternative has been to also weigh the ICD-10 codes according to their distance, this time using a score dependent on the actual distance of the report (e.g., score $= 1/distance$).
The ICD-10 code which receives the highest score after evaluating all reports in the shortlist will be our best guess, but we also compared our second and third best guesses to see if they match the ICD-10 code of our reports.

Chapter 7

Implementation

The actual implementation is done in two parts. The first part is written in the Perl programming language, the second in the C++ language. Both of these programs are developed on a PC with SUSE Linux version 10.1. Following is a description of what both these programs do. The full source code can be found at [21]. The Perl part is used to calculate the distance between reports using all described definitions of distance. It calculates all seven distance between each possible pair of reports and puts these in a separate matrix for each distance. These distance matrices are meant as input for the C++ program. The two programs therefore work together as a separate front- and back-end.

7.1 Perl program

The Perl program is the first step in interpreting the medical reports. It takes a comma separated file with all the reports as input. The input file contains three columns: Patient number, ICD-10 code and the actual written report. It goes through the file one line at a time. The patient number is used to check for duplicate entries, and also because the input file uses a new line when adding an ICD-10 code to the same report (where the patient number is the same, but the report column empty). We store the ICD-10 code and further analyze the written report. The third column containing the actual report consist of three different paragraphs: Clinical information, findings and conclusion. For each of these three paragraphs the words are split and are counted and stored in our document vector. The word count is also stored for the report as a whole as well as the frequency in all reports. When counting words we also immediately store its position for the position distance, we keep track of the distance between the current occurrence and

the last one and if they are within the worddistance-parameter we store them as one interval for the overlap distance. The ICD-10 codes are also saved as a list with all the reports belonging to a certain ICD-10 code. The total frequency count for each ICD-10 code is stored as well. The final action is the storage of a list with ICD-10 codes and the corresponding ICD-10 region, to be able to do the match on the ICD-10 region in the British reports.
Now that we have all the information we need from our input files, we are going to process this information. For each pair of document vectors all seven different distances are calculated and stored in matrices, which are later stored in seven separate files. We only use the bottom half of the matrix, because all our distance definitions result in symmetrical results, i.e., the distance from report A to report B is the same as the distance from report B to report A.

7.2 C++ program

The C++ program takes all seven files containing the distance matrices as its input, as well as the list of ICD-10 codes per report to allow us to check whether the correct ICD-10 code is found. Each of the seven distance matrices are are stored into their respective arrays, and then we perform the selection of the k nearest neighbors. Once we have found those, we are going to use each of our three scoring methods on their ICD-10 codes: most frequently used, fixed score and score relative to the distance. This results in a ranked list of potential ICD-10 codes for each report. We compare the top guess, as well as the top 3, to the actual ICD-10 code of the report. We calculate not only the accuracy but also the true positives, false positives and false negatives for each ICD-10 code which allows us to calculate precision, recall and F-score.

Chapter 8

Results

The results of our program, applied to both datasets, will be explained in this chapter. Note that the heterogeneous dataset is classified in two ways, namely on ICD-10 code but also on ICD-10 region (as has been explained in Section 5.1). This gives us three different results to discuss. In each of them we are going to look for the most probable class, but we will also look at the list of the 3 most probable classes.
In order to be able to analyse the quality of the results of our program we use the leave-one-out principle. This principle generates the results by removing a single report from our data, and use all the other reports as training data. When the classification is complete we compare the outcome with the actual class that was given to the report. We do this for every report. All the reports in our database have their topics manually defined by human experts. The leave-one-out principle ignores the topic of the report being looked at, and compares only the content of the reports. It therefore simulates it as being a new report.
The first thing we look for is the *accuracy* of our program. Accuracy gives the percentage of reports for which our program generates the correct class. Although accuracy is a good way to get a general idea of the performance of the program, we would also like to get an idea about the *recall* and *precision* of our program. A drawback of these measurements is that they are defined per topic only. Thus they give more detailed information about the behavior of our program but it is harder to use them to get a good overview of the classification over all classes. We therefore use the accuracy to find out which combination of methods gives good results and than we are going to use recall and precision to get a more in-depth view.
The precision of a certain topic is the number of true positives (i.e., the number of reports correctly labeled as having that topic) divided by the number of reports labeled as belonging to the topic (i.e., the sum of true

positives and false positives, which are items incorrectly labeled as having that topic). Recall is the number of true positives divided by the number of true positives and false negatives (i.e., reports that were not labeled as having this topic, but should have been). Both recall and precision will therefore have a value between 0 and 1. The definitions of precision and recall are:

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

Here NP = the number of true positives, FP = the number of false positives, and FN = the number of false negatives

As an example, if we define all reports as belonging to topic T, then the recall of T would be 1 (all reports on topic T are returned as topic T), but precision would be close to 0 (many report are incorrectly labeled as belonging to T). And if we defined only 1 report, of which are absolutely sure it belongs to U, as belonging to class U, while there may have been many reports which belong to class U, the precision would be 1 (all reports labeled as U did indeed belong to U) and the recall very low (many reports that belong to class U are not labeled as U).
It is therefore quite clear that recall and precision are not very usable when they are being looked at separately, therefore we use the so-called F-score (or F-measure). This is the harmonic average of precision and recall:

$$F = \frac{2 * precision * recall}{precision + recall}$$

The F-score is the most common measure in Information Retrieval [22]. If we are more precise we have to talk about F_1-score because precision and recall are weighted equally, as opposed to F_2-score which weights recall twice as much as precision, and $F_{0.5}$ which weights precision twice as much as recall.

8.1 Heterogeneous dataset

Because of the relative small number of reports (380) in our heterogeneous dataset and the high number of possible ICD-10 codes (thousands) we are looking at very widely distributed data. Because of the nature of the methods we use we do not expect to get very high accuracy here. Therefore we are not only going to predict the possible ICD-10 codes but we will also be looking at the region of the disease.

8.1.1 The ICD-10 code case

As said before this will be a very tough challenge for our program. When we look at our most probable prediction for each report, in the best case scenario, we are able to find the right ICD-10 code in 23.4% of the cases. This scenario is achieved when $k = 10$, using the weight relative to the distance of the reports, and the multiset1 distance definition. See Table B.1 in Appendix B for an overview of the results from the other methods.
When comparing the top 3 of most probable ICD-10 codes, the correct class was within this top 3, in the best possible solution, in 31.8% of the cases. This happens when $k = 25$, the scoring method is fixed and using the Jaccard distance. See Table B.2 for full results. Both of these results are as expected, due to the small amount of reports and the large amount of possible classes it is indeed hard to classify many reports correctly.
To have a better idea how good the classification really is we have to look at the precision, recall and F-score. We have selected the five classes with the highest frequency. In Table 8.1 are the results of our program, when we only give our best possible guess. In Table 8.2 are the results when we compare our three most probable classes against the actual class in the report. The numbers between brackets in the first column are the number of reports in each respective class. The accuracy when comparing with the top 3 was significantly better than when comparing only the best guess. This makes sense because we accept it as a good guess when any of the classes in the top 3 occurs in the ICD-10 code list of the report. It is therefore remarkable that the recall, precision and F-score between these two methods are similar.

8.1.2 The ICD-10 region case

Because the number of ICD-10 codes is so large we are also classifying the reports on the ICD-10 region. Because the number of regions is smaller than the number of ICD-10 codes we are expecting the results to improve. When we look at Table B.3, we can conclude that the results have indeed improved. When we only look at the prediction that comes out on top of our list we see that the accuracy goes up to 63.6%. If we compare our top 3, we have the correct answer in 72.7% of the cases, see Table B.4 for all the results when using the top 3. From both these tables it is clear that the accuracy is quite a bit better than the accuracy we achieved when searching for the ICD-10 codes.
In the ICD-10 code case we saw that the difference between best guess and the top 3 was marginal. We did the same comparison for the ICD-10 region case. The results can be found in 8.3 and 8.4, for best guess and top 3

ICD-10 code	recall	precision	F-score
I26(39)	0.485	0.943	0.641
J98.1(19)	0.143	0.083	0.105
R65(16)	0.227	0.312	0.263
J90(16)	0.000	0.000	0.000
K80.2(11)	0.273	0.500	0.353

Table 8.1: Results from the heterogeneous dataset, when classifying ICD-10 codes and comparing our best guess. These results are retrieved when $k = 10$, using the relative distance scoring method and multiset1 distance.

ICD-10 code	recall	precision	F-score
I26(39)	0.493	0.974	0.655
J98.1(19)	0.105	0.250	0.148
R65(16)	0.263	0.333	0.294
J90(16)	0.143	0.091	0.111
K80.2(11)	0.222	0.400	0.286

Table 8.2: Results from the heterogeneous dataset, when classifying ICD-10 codes and comparing it with our top 3. These results are retrieved when $k = 25$, using the fixed scoring method and Jaccard distance.

respectively. The results are once again very clear, there is no significant difference between the best guess and the comparison with the three most probable classes.
To be able to make a fair comparison between the classification on ICD-10 code and on ICD-10 region it is necessary to take a closer look at these predictions to see why they have improved, by analysing precision, recall and F-score. When we compare the results from Table 8.1 and Table 8.3, which are both done on the most probable class, and we look at the F-score it becomes immediately clear why the ICD-10 region classification works much better than the ICD-10 code classification. In the lower frequency classes which we see in Table 8.1 we never achieve a proper F-score. Apparently we need more than a few documents in a class to be able to classify the reports belonging to that class correctly. The results in Table 8.3 confirm this assumption. It also shows that both precision and recall are high for the most frequent classes. This means the classification not only labels the reports correctly, but also that it retrieves most reports which do belong to that class.

region	recall	precision	F-score
lung(154)	0.699	0.926	0.797
brain(84)	0.821	0.970	0.889
kidney(56)	0.761	0.795	0.778
abdomen(31)	0.350	1.000	0.519
intestines(26)	0.545	0.571	0.558

Table 8.3: Results from the heterogeneous dataset, when classifying ICD-10 regions and comparing our best guess. These results are retrieved when k = 25, using the fixed distance scoring method and multiset1 distance.

region	recall	precision	F-score
lung(154)	0.762	0.961	0.850
brain(84)	0.896	0.952	0.923
kidney(56)	0.700	0.833	0.761
abdomen(31)	0.417	0.769	0.541
intestines(26)	0.444	0.471	0.457

Table 8.4: Results from the heterogeneous dataset, when classifying ICD-10 regions and comparing it with our top 3. These results are retrieved when $k = 10$, using the fixed distance scoring method and Jaccard distance.

8.2 Homogeneous dataset

The homogeneous dataset contains 441 different reports. We classify these reports on ICD-10 code only because they all deal with the same region. Once again we look at our most probable class, as well as the three most probable classes. In the former (full results can be found in Table B.5 in Appendix B) we achieve an accuracy of up to 36.3% by using the 25 nearest neighbors, relative distance scoring method and the Jaccard distance definition. In the latter (full results in Table B.6) we achieve 57.6% if we look at the 100 nearest neighbors, relative distance scoring and the Jaccard distance.

Compared to the results from the heterogeneous dataset the most notable change is the rather large improvement between the accuracy when looking at the best guess and when looking at the top 3. We know that these reports deal with a very specific disease and location of research, namely acute appendicitis. A possible reason for the big jump in accuracy could be that the classes in the homogeneous dataset are more overlapping, i.e., the reports are more similar and it is harder to distinguish the boundaries between the

different classes.
If we look at the recall, precision and F-score of the best case scenario (see Table 8.5) it is quite clear that in the higher frequency classes the precision is high but recall low while in the lower frequency classes it is exactly the other way around, the recall even reaches the perfect score of 1, but the precision drops accordingly. Having a high precision and low recall, as is the case in the higher frequency range, means that these classes are underestimated: most reports labeled as belonging to these classes do indeed belong to the class, but many are not found. Having high recall and low precision means the class is overestimated: all reports in the class are found, but many others are labeled as belonging to that class as well.

ICD-10 code	recall	precision	F-score
NSAP(90)	0.401	0.856	0.546
K35.9(81)	0.505	0.806	0.621
K35.0(65)	0.333	0.778	0.467
K35.1(56)	0.556	0.147	0.233
K52.9(38)	0.500	0.029	0.056
I88.0(19)	1.000	0.222	0.364
K57(17)	1.000	0.118	0.211

Table 8.5: Results from the homogeneous dataset, when classifying ICD-10 codes and comparing it with our top 3. These results are retrieved when $k = 100$, using the relative distance scoring method and Jaccard distance.

This table only shows the results when comparing the class with our top 3 of most probable classes. The results when comparing the class to our best guess are similar, just like we have seen with the heterogeneous dataset.
These findings support our idea that the documents in the homogeneous dataset are so similar that it is hard to distinguish between one class or the other and as a result the low frequency classes are overestimated and the high frequency classes underestimated.

Chapter 9

Conclusions and future work

During this research we have explored many different options for solving the text classification problem for our two datasets with medical reports. This paper concludes with a structured analysis of each of the methods used and gives suggestions for future research. We will go over each of the different distance measures, the selection using k-Nearest Neighbor, and the different scoring methods used.

Distance measures

The Hamming distance can be seen as the simplest way to define distance, it uses only the number of unique words that two reports differ. In all given circumstances this gives significantly worse results than the other distance measures.

The Jaccard distance uses the same distance as Hamming, but it adds to it a normalization on the number of unique words that two reports share. From the result tables in Appendix B it immediately shows that this is a big improvement over the Hamming distance. The reason why the Jaccard distance performs better is because it not only counts the difference between two reports, but does this relative to the length of the reports, thereby making it a better method to compare our reports, which vary in length from roughly 50 to 400 words.

In the multiset distances we also look at the number of occurrences of a word within a report, whereas Hamming and Jaccard only look at the binary case. Results from both multiset distances are similar to those of the Jaccard distance, i.e., they are within a few percent points of each other, sometimes favoring one of the multiset distances and sometimes the Jaccard distance. We can conclude from this that the frequency of a word does not improve the quality of classification. If we use the multiset distance but give a normal set as its input it generates exactly the Jaccard distance, from this we can conclude that this extra

information does not improve our classification.
The other three relations, the interval relation and both overlap relations, also consider the position of the word occurrences instead of only the frequency count. Although we had high expectations of this approach, it is clear from the results that they are quite a bit worse than the Jaccard and multiset approaches, and more in line with the results from the Hamming distance. The reason for this probably lies in the fact that the medical reports are a summation of what the doctor observes, within these documents there is no standardized order in which these observations have to be written down. The text order is therefore very dependent on the doctor that has written the report and the order in which he looks at the different images from the CT scan. This explains why our results using this approach were not as we expected.

Selection and scoring methods

The choice between the two proposed selection methods, k-Nearest Neighbor and the distance boundary method, was easily made. After the analysis of the results from our first run we already expected k-Nearest Neighbor to have the upper hand and all following runs confirmed this. Having a pre-set distance boundary is not flexible enough to deal with the different high- and low-density areas in the feature vector space. This results in large differences in the amount of documents which are being compared between different areas.
The selection which k is optimal in the k-Nearest Neighbor method, is very challenging for each application that uses this method. We generally found that k should be at least 10 and not more then 50. This number mainly depends on the size of the different classes as well as the total number of reports in a dataset.
To be able to draw conclusions which of the three scoring methods, used to weigh the ICD-10 codes of those reports selected by k-Nearest Neighbor, is superior to the others, we are required to take an in-depth look at the results. We have to look at the two datasets separately, because they give different outcomes. When we look at the homogeneous dataset, which are about patients with the same kind of disease and therefore the reports being very similar, there is a clear winner. The scoring method which uses a score relative to the distance of the report surpasses the other two methods. In the reports from the heterogeneous dataset there is no clear distinction between the relative distance and the fixed distance scoring method. Both are better than the most frequently used score, which gives an equal score to each of the selected

reports, irrespective of their distance or ranking. From this we can conclude that we have to take the (relative) distance of the report into account when weighing the different topics of the nearest reports.

The two datasets

When we consider the entire process of text classification for both datasets there are some very clear similarities as well as some differences. Let us look at the similarities first. Both datasets contain several hundreds reports, which is very few for text mining. Due to this scarcity we did not do any preselection on the reports, which in turn led to datasets which contained ICD-10 codes which would be impossible to find by the techniques we used, or any technique one can think of, simply due to the fact that there was not enough training data for those classes.

If we consider the reports from the heterogeneous dataset then it is clear from looking at the wide spread distribution of the ICD-10 codes that classification is going to be very hard. It is encouraging to see that the most frequent classes achieve decent F-scores. If we classify them on ICD-10 region, the result becomes even better, both in terms of accuracy but mainly when looking at the different F-scores. The F-score gives us the best idea of how well the program classifies the different reports as it incorporates both precision and recall.

For the homogeneous dataset this is a little different. When we compare accuracy, the classification on ICD-10 code appears to be better then it is in the heterogeneous dataset, however there are fewer classes to be considered. When looking closer at the results it becomes clear that the classification is actually worse than it is for the heterogeneous dataset. This can be concluded from the F-score, which is significantly lower, as well as the larger difference between the best guess and when the best three are considered. It is remarkable that the homogeneous dataset performs better when k is larger, compared to the heterogeneous dataset, while the total number of reports is equal. The reason for this is that the reports are more similar on one side, and the classes being larger on the other side.

We have tried to classify the medical reports, using a simple concept but with some elegant ideas and solutions. Using a bag-of-words approach we found the best methods to define the distance to be the Jaccard distance and both multiset distances. Using k-Nearest Neighbor to select the relevant reports and then weighing their topics according to their (relative) distance from the new report, to determine the class for the new report gave the best results. The results in the heterogeneous dataset when classifying on ICD-10

region are especially encouraging. Results of around and over two-third correct are generally accepted to be good results for automated classification of medical reports. Judging from these results we believe the used methods are still viable in a research domain that continues to grow in quantity as well as complexity all the time. Having control over each aspect of the process allowed us to test many different concepts.
The new ideas we developed to also take the position of a word into account have not led to the expected improvement. It would be interesting to do more research on these new techniques. It would probably require a quite organized, hierarchical structure of the document to successfully deploy these ideas. To be able to test our methods beyond that of these two dataset it would be interesting to do the same on a larger dataset, which would also allow us to do some preselection of reports and classes. The lack of a standard dataset with medical reports which is freely available makes it very hard to compare results between different researchstudies. Most, if not all, do a form of preselection which obviously can have a major impact on the results.
Other research indicates that classification of medical reports is very dependent on the input of knowledge from a domain expert. Another suggestion for future work would be to try and extract this knowledge from the already classified documents. While we only compare our reports on the basis of how similar they are it might be worth trying to discover the relationship between certain features and the corresponding ICD-10 code. This could lead to a large increase in correctly classified reports.

Appendix A

ICD-10 categories

Chapter	Blocks	Title
I	A00-B99	Certain infectious and parasitic diseases
II	C00-D48	Neoplasms
III	D50-D89	Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism
IV	E00-E90	Endocrine, nutritional and metabolic diseases
V	F00-F99	Mental and behavioural disorders
VI	G00-G99	Diseases of the nervous system
VII	H00-H59	Diseases of the eye and adnexa
VIII	H60-H95	Diseases of the ear and mastoid process
IX	I00-I99	Diseases of the circulatory system
X	J00-J99	Diseases of the respiratory system
XI	K00-K93	Diseases of the digestive system
XII	L00-L99	Diseases of the skin and subcutaneous tissue
XIII	M00-M99	Diseases of the musculoskeletal system and connective tissue
XIV	N00-N99	Diseases of the genitourinary system
XV	O00-O99	Pregnancy, childbirth and the puerperium
XVI	P00-P96	Certain conditions originating in the perinatal period
XVII	Q00-Q99	Congenital malformations, deformations and chromosomal abnormalities
XVIII	R00-R99	Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified
XIX	S00-T98	Injury, poisoning and certain other consequences of external causes
XX	V01-Y98	External causes of morbidity and mortality
XXI	Z00-Z99	Factors influencing health status and contact with health services
XXII	U00-U99	Codes for special purposes

Table A.1: ICD-10 Classification.

Appendix B

Result tables

In this appendix all the full result tables as discussed in Chapter 8 and 9 are presented.
In these tables we use the following abbreviations:

- k: The number of closest reports we consider.
- mfu: Most frequently used. Each topic in the shortlist is counted and the most frequent one is chosen as the topic of the new report.
- fixed: The fixed scoring method, where each topic receives a score of $\{k, k-1, k-2, \ldots, 3, 2, 1\}$ in order of increasing distance.
- distance: The scoring method where topics receive a score of $1/distance$.

	Hamming	Jaccard	interval	overlap1	overlap2	multiset1	multiset2
(k=3),mfu	9.2%	17.9%	8.9%	8.9%	7.9%	17.6%	16.8%
(k=3),fixed	5.0%	18.7%	4.2%	4.2%	2.4%	18.7%	19.2%
(k=3),distance	5.8%	18.9%	3.9%	3.9%	2.6%	18.4%	18.7%
(k=5),mfu	12.1%	20.3%	11.6%	11.6%	9.2%	19.7%	19.5%
(k=5),fixed	7.1%	20.8%	6.1%	6.1%	5.0%	19.7%	19.7%
(k=5),distance	7.9%	21.1%	6.8%	6.8%	6.1%	20.8%	21.1%
(k=10),mfu	12.9%	22.1%	10.3%	10.3%	9.5%	21.3%	20.3%
(k=10),fixed	11.1%	21.3%	8.7%	8.9%	7.9%	21.6%	21.1%
(k=10),distance	12.1%	21.8%	8.4%	8.4%	8.2%	23.4%	22.1%
(k=25),mfu	13.7%	17.9%	13.7%	13.7%	13.2%	17.6%	17.9%
(k=25),fixed	15.0%	22.4%	13.2%	13.2%	12.1%	21.1%	21.3%
(k=25),distance	14.7%	18.9%	14.7%	14.7%	15.3%	19.5%	19.5%
(k=100),mfu	10.3%	15.0%	10.3%	10.3%	10.3%	13.2%	13.7%
(k=100),fixed	13.4%	18.4%	12.6%	12.6%	12.6%	16.3%	15.8%
(k=100),distance	10.8%	16.6%	10.8%	10.8%	10.8%	13.9%	14.2%

Table B.1: Accuracy for the reports in the heterogeneous dataset for finding the ICD-10 code, when looking at the most probable prediction only (380 reports).

	Hamming	Jaccard	interval	overlap1	overlap2	multiset1	multiset2
(k=3),mfu	9.5%	19.7%	8.9%	8.9%	7.9%	19.2%	18.2%
(k=3),fixed	11.1%	25.3%	10.3%	10.3%	8.4%	26.1%	25.5%
(k=3),distance	10.5%	25.8%	10.0%	10.0%	8.7%	26.1%	25.5%
(k=5),mfu	12.6%	21.6%	11.6%	11.6%	9.2%	21.8%	22.1%
(k=5),fixed	13.4%	26.6%	13.4%	13.4%	11.6%	26.1%	26.3%
(k=5),distance	13.7%	27.6%	13.4%	13.4%	11.6%	27.1%	27.9%
(k=10),mfu	13.9%	25.0%	11.1%	11.1%	10.0%	24.7%	23.9%
(k=10),fixed	15.3%	29.7%	13.9%	13.9%	12.9%	28.9%	28.2%
(k=10),distance	15.8%	31.1%	13.4%	13.4%	12.9%	30.0%	28.7%
(k=25),mfu	15.3%	24.7%	15.3%	15.3%	14.7%	24.5%	25.3%
(k=25),fixed	20.0%	31.8%	18.9%	18.9%	17.4%	29.5%	28.7%
(k=25),distance	20.0%	28.7%	20.0%	20.0%	20.0%	28.4%	29.7%
(k=100),mfu	13.4%	18.2%	13.2%	13.2%	13.2%	17.4%	17.4%
(k=100),fixed	16.6%	25.8%	15.8%	15.8%	15.8%	23.7%	24.5%
(k=100),distance	14.5%	21.3%	14.5%	14.5%	13.7%	18.7%	20.5%

Table B.2: Accuracy for the reports in the heterogeneous dataset for finding the ICD-10 code, when looking at the top 3 of most probable predictions (380 reports).

	Hamming	Jaccard	interval	overlap1	overlap2	multiset1	multiset2
(k=3),mfu	29.8%	61.2%	30.0%	29.5%	25.9%	60.6%	60.3%
(k=3),fixed	22.6%	61.4%	21.8%	21.8%	19.3%	58.7%	59.0%
(k=3),distance	29.8%	62.8%	29.2%	28.9%	26.7%	59.8%	60.3%
(k=5),mfu	37.5%	60.9%	32.2%	32.2%	28.1%	61.7%	61.2%
(k=5),fixed	35.8%	62.8%	32.0%	32.0%	29.8%	62.5%	62.5%
(k=5),distance	38.8%	63.4%	33.3%	33.3%	30.6%	63.1%	63.1%
(k=10),mfu	40.2%	62.3%	35.0%	35.0%	31.4%	62.8%	63.1%
(k=10),fixed	43.3%	62.8%	37.2%	37.2%	33.1%	62.0%	63.1%
(k=10),distance	43.3%	62.8%	37.2%	37.2%	33.3%	62.3%	63.4%
(k=25),mfu	45.2%	61.2%	41.0%	41.0%	39.4%	61.4%	62.0%
(k=25),fixed	43.5%	62.3%	38.0%	37.7%	34.4%	63.6%	63.4%
(k=25),distance	46.3%	61.7%	43.0%	43.0%	40.5%	62.5%	62.5%
(k=100),mfu	43.8%	54.8%	43.8%	43.8%	43.8%	53.2%	54.3%
(k=100),fixed	45.5%	59.5%	41.9%	41.9%	39.7%	58.4%	58.7%
(k=100),distance	44.9%	55.1%	44.6%	44.6%	44.9%	53.4%	54.8%

Table B.3: Accuracy for the reports in the heterogeneous dataset for finding the ICD-10 region, when looking at the most probable prediction only (380 reports).

	Hamming	Jaccard	interval	overlap1	overlap2	multiset1	multiset2
(k=3),mfu	30.9%	65.3%	30.6%	30.0%	26.4%	63.9%	63.4%
(k=3),fixed	26.4%	67.5%	24.8%	24.5%	21.2%	65.8%	65.8%
(k=3),distance	33.6%	68.9%	32.5%	32.0%	28.4%	67.5%	67.5%
(k=5),mfu	39.9%	66.9%	35.0%	35.3%	30.6%	65.8%	65.3%
(k=5),fixed	41.3%	71.3%	36.6%	36.6%	33.3%	68.6%	68.3%
(k=5),distance	44.1%	71.9%	38.6%	38.8%	34.7%	69.4%	68.9%
(k=10),mfu	43.0%	69.1%	39.1%	39.1%	35.8%	68.0%	67.5%
(k=10),fixed	48.2%	72.7%	43.5%	43.3%	37.5%	68.6%	69.1%
(k=10),distance	47.9%	72.2%	42.1%	42.1%	37.7%	69.4%	69.4%
(k=25),mfu	51.2%	70.8%	47.4%	47.4%	44.6%	68.6%	68.0%
(k=25),fixed	49.6%	71.6%	44.9%	44.6%	40.5%	69.7%	69.4%
(k=25),distance	52.6%	72.5%	49.6%	49.6%	45.7%	70.5%	69.4%
(k=100),mfu	53.2%	66.7%	53.4%	53.4%	54.0%	65.0%	65.6%
(k=100),fixed	54.8%	70.2%	51.5%	51.5%	49.6%	66.4%	66.4%
(k=100),distance	53.4%	68.0%	53.2%	53.2%	53.7%	65.6%	66.1%

Table B.4: Accuracy for the reports in the heterogeneous dataset for finding the ICD-10 region, when looking at the top 3 of most probable predictions (380 reports).

	Hamming	Jaccard	interval	overlap1	overlap2	multiset1	multiset2
(k=3),mfu	18.4%	27.9%	17.5%	17.7%	16.6%	26.5%	27.2%
(k=3),fixed	22.7%	30.8%	23.8%	23.8%	22.9%	29.7%	29.0%
(k=3),distance	23.1%	32.2%	22.9%	23.1%	21.8%	30.8%	31.3%
(k=5),mfu	17.0%	27.9%	15.9%	15.9%	15.6%	28.6%	28.1%
(k=5),fixed	21.8%	29.3%	19.5%	20.0%	19.0%	30.2%	30.4%
(k=5),distance	22.0%	32.4%	21.8%	21.8%	22.7%	32.2%	32.4%
(k=10),mfu	21.5%	26.8%	18.1%	18.1%	20.9%	26.3%	26.8%
(k=10),fixed	20.9%	28.6%	21.1%	20.9%	20.4%	29.0%	29.3%
(k=10),distance	26.1%	32.7%	24.9%	24.9%	26.8%	33.8%	34.7%
(k=25),mfu	26.3%	28.8%	26.5%	26.3%	25.2%	26.1%	28.1%
(k=25),fixed	25.2%	31.1%	24.7%	24.5%	25.2%	31.1%	29.0%
(k=25),distance	28.8%	36.3%	31.1%	30.8%	29.7%	31.7%	33.1%
(k=100),mfu	23.8%	30.6%	23.8%	23.8%	24.3%	30.2%	29.9%
(k=100),fixed	26.8%	30.2%	26.3%	26.3%	26.5%	32.0%	30.8%
(k=100),distance	25.4%	33.6%	27.2%	27.2%	27.0%	34.0%	32.9%

Table B.5: Accuracy for the reports in the homogeneous dataset for finding the ICD-10 code, when looking at the most probable prediction only (441 reports).

	Hamming	Jaccard	interval	overlap1	overlap2	multiset1	multiset2
(k=3),mfu	22.2%	32.4%	21.5%	21.3%	20.2%	31.1%	31.1%
(k=3),fixed	26.3%	38.8%	26.3%	26.3%	24.7%	39.2%	39.9%
(k=3),distance	29.0%	42.0%	28.6%	28.3%	26.8%	42.0%	42.9%
(k=5),mfu	25.2%	36.5%	24.0%	24.0%	23.6%	35.6%	37.2%
(k=5),fixed	29.5%	42.4%	27.2%	27.2%	24.9%	43.8%	44.2%
(k=5),distance	32.4%	45.6%	30.2%	30.2%	30.4%	45.4%	46.3%
(k=10),mfu	32.4%	38.5%	30.2%	29.9%	32.9%	38.3%	39.0%
(k=10),fixed	32.2%	44.7%	30.8%	30.6%	29.9%	43.1%	43.3%
(k=10),distance	37.4%	47.4%	36.1%	36.1%	39.0%	46.5%	49.0%
(k=25),mfu	44.4%	46.5%	43.8%	43.5%	42.0%	43.5%	46.3%
(k=25),fixed	41.0%	47.8%	41.5%	41.3%	40.4%	46.9%	46.7%
(k=25),distance	46.9%	53.5%	47.6%	47.4%	45.8%	51.0%	52.2%
(k=100),mfu	46.5%	53.7%	48.5%	48.3%	47.2%	52.4%	52.2%
(k=100),fixed	49.7%	54.6%	47.4%	47.4%	47.6%	53.7%	54.6%
(k=100),distance	48.5%	57.6%	51.5%	51.2%	51.0%	55.3%	54.9%

Table B.6: Accuracy for the reports in the homogeneous dataset for finding the ICD-10 code, when looking at the top 3 of most probable predictions (441 reports).

Bibliography

[1] Stuart J. Russell and Peter Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, third edition, 2010.

[2] Alan M. Turing. Computing machinery and intelligence. *Mind, New Series*, 59(236):433–460, October 1950.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, second edition, 2009.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 2006.

[6] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[7] Ronen Feldman and Ido Dagan. Knowledge discovery in textual databases. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 112–117, 1995.

[8] Prabhakar Raghavan. Extracting and exploiting structure in text search. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, 2003. Industrial paper.

[9] Christos Faloutsos and Douglas W. Oard. A survey of information retrieval and filtering methods. Computer Science Technical Report Series; Vol. CS-TR-3514, 1995.

[10] Aurangzeb Khan, Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology*, 1(1):4–20, February 2010.

[11] Eui-Hong Han, George Karypis, and Vipin Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In *Proceedings of the 5th Pacific-Asia Knowledge Discovery and Data Mining Conference*, pages 53–65, 2001.

[12] World Health Organization. International Classification of Diseases. Website. `http://www.who.int/classifications/icd/en/`. Retrieved August 8, 2010.

[13] Stanley S. Stevens. On the theory of scales of measurement. *Science*, 103(2684):677–680, 1946.

[14] Adam B. Wilcox, George Hripcsak, and Carol Friedman. Using knowledge sources to improve classification of medical text reports. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 116–117, 2000.

[15] Adam B. Wilcox and George Hripcsak. Research paper: The role of domain knowledge in automating medical text report classification. *Journal of the American Medical Informatics Association*, 10(4):330–338, 2003.

[16] John P. Pestian, Christopher Brew, Pawel Matykiewicz, D. J. Hovermale, Neil Johnson, K. Bretonnel Cohen, and Wlodzislaw Duch. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP*, pages 97–104, 2007.

[17] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[18] Ruben van Bodegom. Automated text categorization of Reuters newsreports. Researchproject, Leiden University. `http://www.liacs.nl/~rvbodego/researchproject/researchproject.pdf`. Retrieved August 8, 2010.

[19] Planetmath. Website. `http://planetmath.org/encyclopedia/Multiset.html`. Retrieved August 8, 2010.

[20] Walter A. Kosters and Jeroen F. J. Laros. Metrics for mining multisets. In *Research and Development in Intelligent Systems XXIV, Proceedings of the Twenty-seventh SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 293–303, 2007.

[21] Ruben van Bodegom. Website. `http://www.liacs.nl/~rvbodego/thesis`. Retrieved August 8, 2010.

[22] Cornelis J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, second edition, 1979.